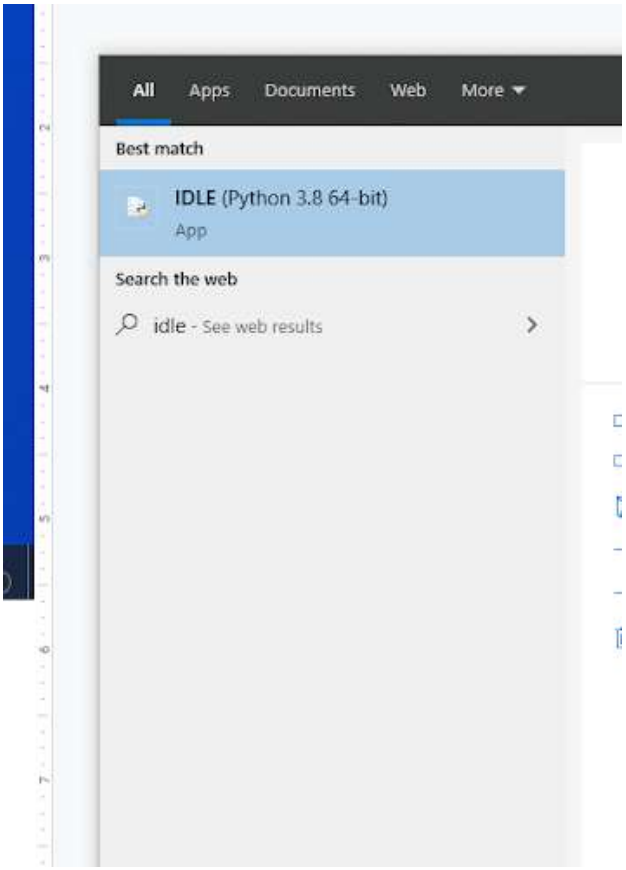
Chapter 9 Practice Directions

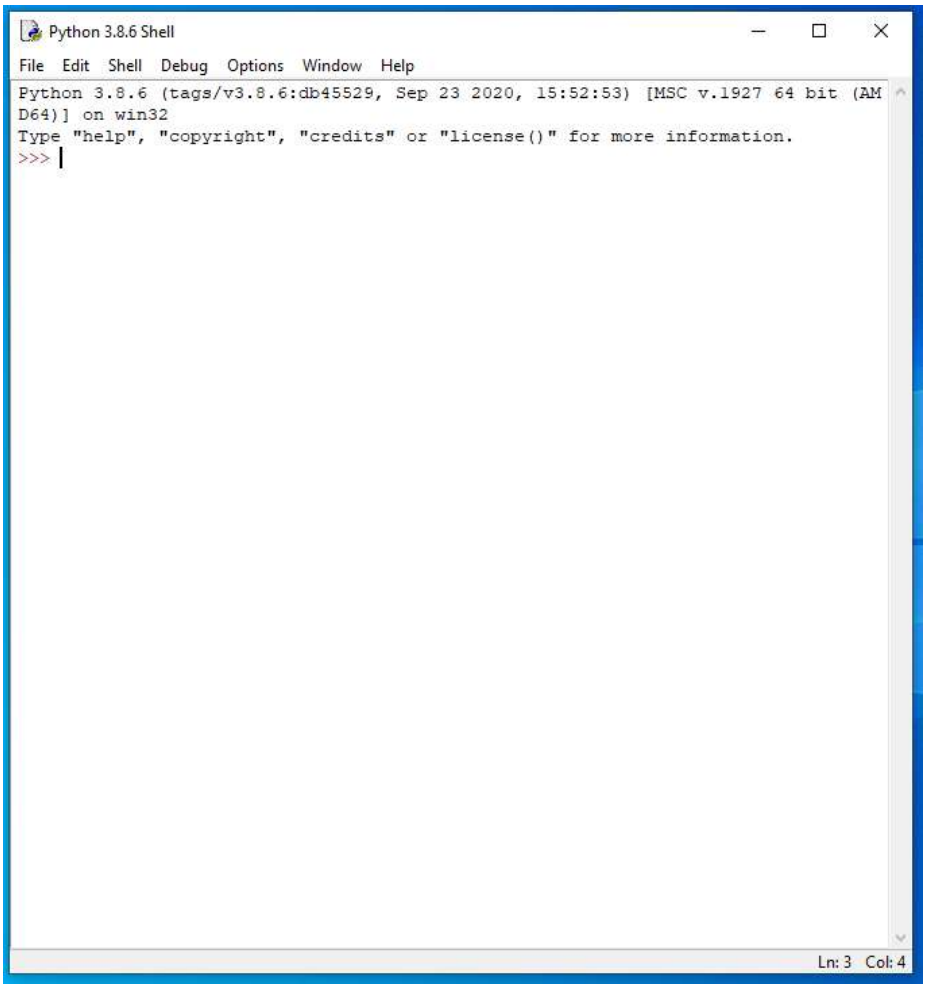
1. . Using your Windows button menu, find and launch your IDLE program.



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.

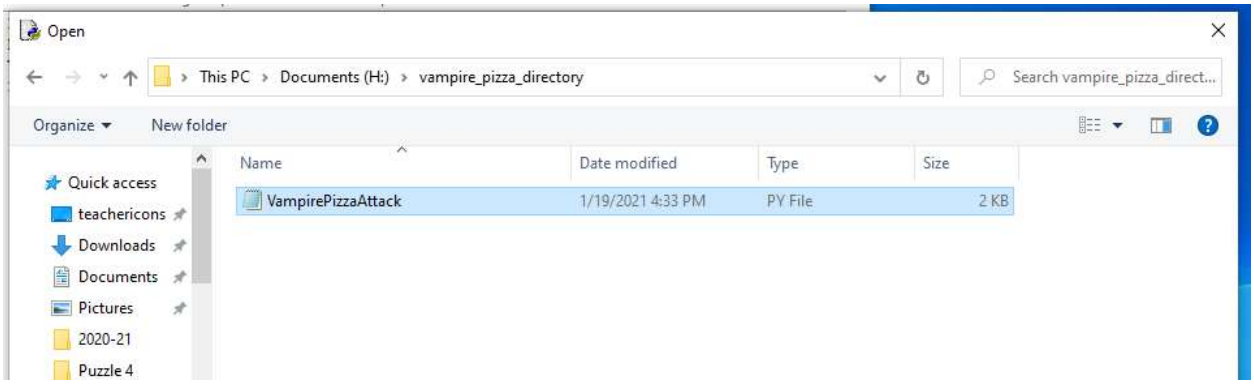
2. Your IDLE window should look something like this once it has launched.:



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 29.

```
#Set up rates
SPAWNRATE = 360
FRAMERATE = 60
```

7. Press ENTER twice.
8. Type the code that you see on Lines 31 – 33 of the screenshot below.

```
#Set up rates
SPAWNRATE = 360
FRAMERATE = 60

#Define speeds
REG_SPEED = 2
SLOW_SPEED = 1
```

Line 31 contains a comment.

Lines 32 and 33 create two constant variables called REG_SPEED and SLOW_SPEED and set their values to 2 and 1, respectively.

9. Scroll down and click at the end of Line 62, after you define the self.speed variable for the enemy sprite objects.

```
    #This function creates an instance of the enemy
    def __init__(self):
        super().__init__()
        self.speed = 2
```

10. Replace the number 2 with the code that you see in the screenshot below.

```
    #This function creates an instance of the enemy
    def __init__(self):
        super().__init__()
        self.speed = REG_SPEED
        self.lane = randint(0, 4)
        all_vampires.add(self)
```

We have simply replaced the value of 2 with the value of whatever the value of the REG_SPEED variable is. Currently, it is set to 2 as well. However, if we decide to change it later, this will make it easier to avoid problems.

11. Scroll down and click at the end of Line 130, as shown in the screenshot below.

```
        #Set up the background tiles to respond to a mouse click
        elif event.type == pygame.MOUSEBUTTONDOWN:
            x, y = pygame.mouse.get_pos()
            tile_grid[y//100][x//100].effect = True
            print(x, y)
            print('You clicked me!')
```

12. Delete the bottom two print statements.

```
        #Set up the background tiles to respond to a mouse click
        elif event.type == pygame.MOUSEBUTTONDOWN:
            x, y = pygame.mouse.get_pos()
            tile_grid[y//100][x//100].effect = True

#-------------------------------------------------------------
```

13. Scroll down and click at the end of Line 140, as shown in the screenshot below.

```
#-------------------------------------------------------------
#Set up collision detection
    #draw a background grid
    for tile_row in tile_grid:
        for tile in tile_row:
            GAME_WINDOW.blit(BACKGROUND, (tile.rect.x, tile.rect.y), tile.rect)
```

14. Press ENTER twice.

15. Type the code that you see on Lines 142 - 148. Make sure your indentation matches what is shown in the screenshot below.

```
#------------------------------------------------------------
#Set up collision detection
    #draw a background grid
    for tile_row in tile_grid:
        for tile in tile_row:
            GAME_WINDOW.blit(BACKGROUND, (tile.rect.x, tile.rect.y), tile.rect)

    for vampire in all_vampires:
        tile_row = tile_grid[vampire.rect.y //100]
        vampire_left_side_x = vampire.rect.x // 100
        vampire_right_side_x = (vampire.rect.x + vampire.rect.width) // 100
        if -1 < vampire_left_side_x < 10:
            left_tile = tile_row[vampire_left_side_x]
        else:
```

The code on Line 142 begins a “for” loop that will iterate through each vampire pizza sprite in the all_vampires list.

The code on Line 143 figures out the row that the vampire sprite it is currently working on is in. It assigns that row’s number to the tile_row variable.

Line 144 creates a variable called vampire_left_side_x. It’s value is set to the current x-location of the vampire sprite’s left edge (the X coordinate of the vampire sprite’s left edge).

Line 145 creates a variable called vampire_right_side_x and sets it’s value to be the current x-location of the vampire sprite’s right edge. You use a math formula to calculate this.

Line 146 creates an if statement that checks to see if the vampire sprite’s left side x coordinate location is between 0 and 9 (or greater than the number -1 and less than the number 10). If it is, it will create a variable called left_tile. If the sprite’s x coordinate is between 0 and 9, it will also calculate the vampire’s column location and the index of the column that is touching the vampire sprite’s left side. It will set that index value to be the value of the left_tile variable. (Line 147)

Line 148 begins your “else” statement, which is what will happen if the current vampire sprite it is looping through is NOT anywhere on the grid.

16. Press ENTER.

17. Type the code that you see on Lines 149 – 153 of the screenshot below.

```
if -1 < vampire_left_side_x < 10:
    left_tile = tile_row[vampire_left_side_x]
else:
    left_tile = None
if -1 < vampire_right_side_x < 10:
    right_tile_wall = tile_row[vampire_right_side_x]
else:
    right_tile = None
```

The code on Line 149 runs if there is no vampire sprite on the grid, meaning that it can't calculate which column the vampire sprite is in. Line 149 creates the left_tile variable and sets it value to None for that current sprite it is looping through.

Lines 150 – 153 run through the same set of steps to find the index value of the tile touching the vampire sprite's right side.

It is necessary to calculate the index values of both the left and right side columns that are touching the vampire sprite because it is possible (when the sprite is all the way at the right edge or left edge of the game window) that the sprite will not have any tiles touching it on one side and it will have a tile touching it on the other. We need to check both sides of our sprite each time we run this loop to check for collisions between sprites.

18. Type the code you see on Lines 154 – 159 of the screenshot below:

```
        left_tile = None
    if -1 < vampire_right_side_x < 10:
        right_tile_wall = tile_row[vampire_right_side_x]
    else:
        right_tile = None
    if bool(left_tile) and left_tile.effect:
        vampire_speed = SLOW_SPEED
    if bool(right_tile) and right_tile.x != left_tile.x and right_tile.effect:
        vampire.speed = SLOW_SPEED
    if vampire.rect.x <= 0:
        vampire.kill()
```

I know this seems like three more “if” statements added at the end of your code, and that’s true – that’s all it is! But it is also a bit more than that. These three lines of code are inherently what make your collision detection in your game work.

Line 154 checks to see if the vampire is touching a tile on its left side (if the index value stored in the left_tile variable is a number of contains the word None – if the left_tile variable is True or False) and, if it is True, if the tile to the left of it has been clicked already (it’s effect). If both of those statements are true, Line 155 will create a vampire_speed variable and set it to be the value of the SLOW_SPEED constant variable that you created earlier.

Lines 156 and 157 will check to see if the vampire sprite is touching a tile on its right side, ensure that the right and the left side x locations are touching different tiles, and check the value of the effect variable for the tile on the sprite’s right side. If all of those conditions return “True” (meaning that the vampire sprite has a tile on its right side, it is different than the tile that is on its left side, and the tile on the right side has been clicked already), Line 157 will create a vampire_speed variable and set its value to be equal to the value of the SLOW_SPEED constant variable you created earlier.

Lines 158 and 159 will track the vampire sprite’s x-location and, when it is less than or equal to 0 (meaning it has moved all the way across the game window), than the enemy vampire sprite will be deleted by executing the kill() method.

19. Go to File > Save Now to save your code.

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
```

```
#Define speeds
REG_SPEED = 2
SLOW_SPEED = 1

#-------------------------------------------------------------
#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 = REG_SPEED
        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 a class of sprites. Each tile has an invisible interactive field attached to it which is a sprite in this class.
class BackgroundTile(sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.effect = False

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

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

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

#Create an empty list to hold the tile grid
tile_grid=[]

#Define the color of the grid outline
tile_color = WHITE

#Populate the background grid
for row in range(6):
    row_of_tiles = []
    tile_grid.append(row_of_tiles)
    for column in range(11):
        new_tile = BackgroundTile()
        new_tile.rect = pygame.Rect(WIDTH * column, HEIGHT * row, WIDTH, HEIGHT)
        row_of_tiles.append(new_tile)
        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:

#-------------------------------------------------------------
#Check for events

    #Checking for and handling events
    for event in pygame.event.get():
        #Exit loop on quit
        if event.type == QUIT:
            game_running = False

        #Set up the background tiles to respond to a mouse click
        elif event.type == pygame.MOUSEBUTTONDOWN:
            x, y = pygame.mouse.get_pos()
            tile_grid[y//100][x//100].effect = True

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

#-------------------------------------------------------------
#Set up collision detection
    #draw a background grid
    for tile_row in tile_grid:
        for tile in tile_row:
            GAME_WINDOW.blit(BACKGROUND, (tile.rect.x, tile.rect.y), tile.rect)
```

```
    for vampire in all_vampires:
        tile_row = tile_grid[vampire.rect.y //100]
        vampire_left_side_x = vampire.rect.x // 100
        vampire_right_side_x = (vampire.rect.x + vampire.rect.width) // 100
        if -1 < vampire_left_side_x < 10:
            left_tile = tile_row[vampire_left_side_x]
        else:
            left_tile = None
        if -1 < vampire_right_side_x < 10:
            right_tile_wall = tile_row[vampire_right_side_x]
        else:
            right_tile = None
        if bool(left_tile) and left_tile.effect:
            vampire_speed = SLOW_SPEED
        if bool(right_tile) and right_tile.x != left_tile.x and right_tile.effect:
            vampire.speed = SLOW_SPEED
        if vampire.rect.x <= 0:
            vampire.kill()

#------------------------------------------------------------
    #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()
```