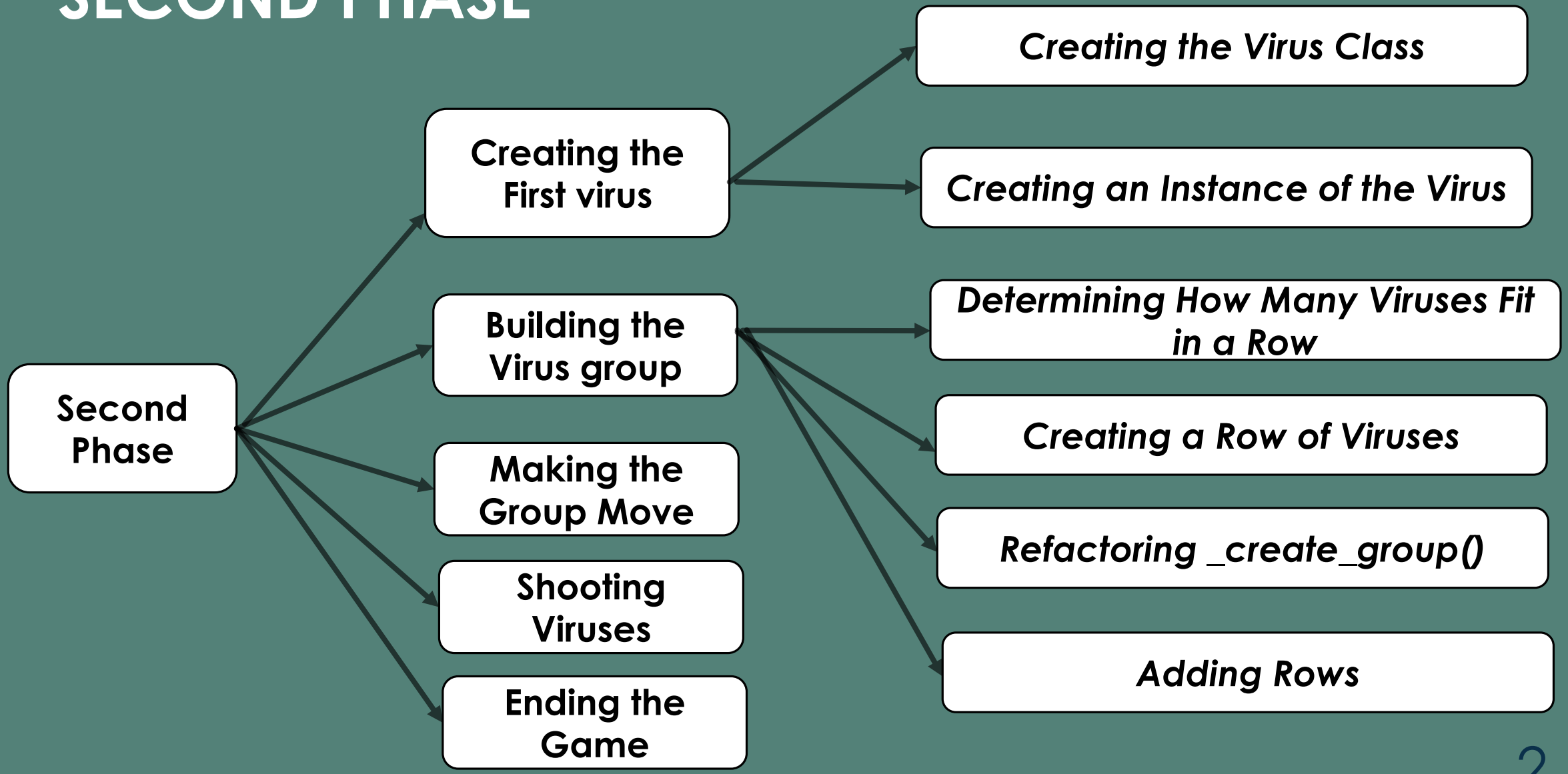
OBJECT ORIENTED PROGRAMMING WITH PYTHON

PROJECT BASED

2nd semester (Lect8)

Dr. Ann Al-Kazzaz

SECOND PHASE



- *Creating the First Virus*
- *Creating an Instance of the Virus*
- Building the Virus group
- *Determining How Many Viruses Fit in a Row*
- *Creating a Row of Viruses*
- *Refactoring _create_group()*
- *Adding rows*

CREATING THE FIRST VIRUS

- Placing one **virus** on the screen is like placing a **robot** on the screen. Each virus's behavior is controlled by a **class** called **Virus**, which we'll structure like the **Robot class**.
- Make sure you save the image file you choose in the *images* folder.

CREATING THE VIRUS CLASS

- Now we'll write the Virus class and save it as ***virus.py***:
- Most of this class is like the Robot class except for the virus' placement on the screen.

Virus.py

```
import pygame
from pygame.sprite import Sprite
class Virus(Sprite):
    """A class to represent a single Virus in the group."""
    def __init__(self, vi_game):
        """Initialize the Virus and set its starting position."""
        super().__init__()
        self.screen = vi_game.screen
        # Load the Virus image and set its rect attribute.
        self.image = pygame.image.load('images/virus.png')
        self.rect = self.image.get_rect()
        # Start each new virus near the top left of the screen.
1       self.rect.x = self.rect.width
        self.rect.y = self.rect.height
        # Store the virus's exact horizontal position.
2       self.x = float(self.rect.x)
```

1. We initially place each virus near the top-left corner of the screen;
   - we add a space to the left of it that's equal to the virus's width and a space above it equal to its height, so it's easy to see.

2. We're mainly concerned with the viruss' horizontal speed, so we'll track the horizontal position of each virus precisely.

- **Note:This Virus class doesn't need a method for drawing it to the screen; instead, we'll use a Pygame group method that automatically draws all the elements of a group to the screen.**

CREATING AN INSTANCE OF THE VIRUS

- We want to create an instance of Virus so we can see the first virus on the screen.
- Because it's part of our setup work, we'll add the code for this instance at the end of the **__init__()** method in **VirusInvasion**
- Finally, we'll create an entire group of viruses, which will be quite a bit of work, so we'll make a new helper method called **_create_group()**.
- Here are the updated import statements for *virus_invasion.py*:

virus_invasion.py

```
--snip--
from laser_beam import Laser_beam
from virus import Virus
```

- And here's the updated __init__() method:

virus_invasion.py

```
def __init__(self):
    --snip--
    self.robot = Robot(self)
    self.laser_beams = pygame.sprite.Group()
    self.viruses = pygame.sprite.Group()
    self._create_group()
```

CREATING AN INSTANCE OF THE VIRUS

- We create a group to hold the group of viruses, and we call **_create_group()**, which we're about to write. Here's the new **_create_group()** method:

virus_invasion.py

```
def _create_group(self):
    """Create the group of viruses."""
    # Make a virus.
    virus = Virus(self)
    self.viruses.add(virus)
```

- In this method, we're creating one instance of Virus, and then adding it to the group that will hold the group.
- The virus will be placed in the default upper-left area of the screen, which is perfect for the first virus.
- To make the virus appear, we need to call the group's **draw()** method in **_update_screen()**:

virus_invasion.py

```
def _update_screen(self):
    --snip--
    for laser_beam in self.laser_beams.sprites():
        laser_beam.draw_laser_beam()
    self.viruses.draw(self.screen)
    pygame.display.flip()
```

CREATING AN INSTANCE OF THE VIRUS

- When you call **draw()** on a group, Pygame draws each element in the group at the position defined by **its rect attribute**. The **draw()** method requires one argument: a surface on which to draw the elements from the group.
- The following figure shows the first virus on the screen.

BUILDING THE VIRUS GROUP

- To draw a group, we need to figure out how many viruses can fit across the screen and how many rows of viruses can fit down the screen. We'll first figure out the horizontal spacing between viruses and create a row; then we'll determine the vertical spacing and create an entire group.

Determining How Many Viruses Fit in a Row

- To figure out how many viruses fit in a row, let's look at how much horizontal space we have. The screen width is stored in **settings.screen_width**, but we need an empty margin on either side of the screen.
- We'll make this margin the width of one virus. Because we have two margins, the available space for viruses is the screen width minus two virus widths:

```
available_space_x = settings.screen_width – (2 * virus_width)
```

DETERMINING HOW MANY VIRUSES FIT IN A ROW

- We also need to set the spacing between viruses;
- **we'll make it one virus width**. The space needed to display one virus is twice its width: one width for the virus and one width for the empty space to its right.
- To find the number of viruses that fit across the screen, we divide the available space by two times the width of a virus.
- We use *floor division* (//), which divides two numbers and drops any remainder, so we'll get an integer number of viruses:

```
number_viruses_x = available_space_x // (2 * virus_width)
```

CREATING A ROW OF VIRUSES

We're ready to generate a full row of viruses. Because our code for making a single virus works, we'll rewrite _create_group() to make a whole row of viruses:

virus_invasion.py

```
def _create_group(self):
    """Create the group of viruses."""
    # Create a virus and find the number of viruses in a row.
    # Spacing between each virus is equal to one virus width.
1   virus = Virus(self)
2   virus_width = virus.rect.width
3   available_space_x = self.settings.screen_width - (2 * virus_width)
    number_viruses_x = available_space_x // (2 * virus_width)
    # Create the first row of viruses.
4   for virus_number in range(number_viruses_x):
        # Create a virus and place it in the row.
        virus = Virus(self)
5       virus.x = virus_width + 2 * virus_width * virus_number
        virus.rect.x = virus.x
        self.viruses.add(virus)
```

CREATING A ROW OF VIRUSES

- We've already thought through most of this code.

1. We need to know the virus's width and height to place viruses, so we create a virus before we perform calculations. This virus won't be part of the group, so don't add it to the group viruses.

2. Here we get the virus's width from its **rect** attribute and store this value in **virus_width** so we don't have to keep working through the **rect** attribute.

3. we calculate the horizontal space available for viruses and the number of viruses that can fit into that space.

4. Next, we set up a loop that counts from 0 to the number of viruses we need to make.

5. In the main body of the loop, we create a new virus and then set its x-coordinate value to place it in the row.

- Each virus is pushed to the right one virus width from the left margin. Next, we multiply the virus width by 2 to account for the space each virus takes up, including the empty space to its right, and we multiply this amount by the virus's position in the row. We use the virus's x attribute to set the position of its rect. Then we add each new virus to the group viruses.

CREATING A ROW OF VIRUSES

When you run *Virus Invasion* now, you should see the first row of viruses appear, as in this Figure

- The first row is offset to the left, which is actually good for gameplay. The reason is that we want the group to move right until it hits the edge of the screen, then drop down a bit, then move left, and so forth.
- this movement is more interesting than having the group drop straight down. We'll continue this motion until all viruses are shot down or until a virus hits the robot or the bottom of the screen.

REFACTORING _CREATE_GROUP()

If the code we've written so far was all we need to create a group, we'd probably leave **_create_group()** as is. But we have more work to do, so let's clean up the method a bit. We'll add a new helper method, **_create_virus()**, and call it from **_create_group()**:

virus_invasion.py

```
def _create_group(self):
    --snip--
    # Create the first row of viruses.
    for virus_number in range(number_viruses_x):
        self._create_virus(virus_number)

def _create_virus(self, virus_number):
    """Create a virus and place it in the row."""
    virus = Virus(self)
    virus_width = virus.rect.width
    virus.x = virus_width + 2 * virus_width * virus_number
    virus.rect.x = virus.x
    self.viruses.add(virus)
```

- The method **_create_virus()** requires one parameter in addition to self: it needs the virus number that's currently being created.
- We use the same body we made for **_create_group()** except that we get the width of a virus inside the method instead of passing it as an argument.
- This refactoring will make it easier to add new rows and create an entire group.

ADDING ROWS

- To finish the group, we'll determine the number of rows that fit on the screen and then repeat the loop for creating the viruses in one row until we have the correct number of rows.
- To determine the number of rows, we find the available vertical space by subtracting the virus height from the top, the robot height from the bottom, and two virus heights from the bottom of the screen:

```
available_space_y = settings.screen_height – (3 * virus_height) – robot_height
```

- The result will create some empty space above the robot, so the player has some time to start shooting viruses at the beginning of each level.
- Each row needs some empty space below it, which we'll make equal to the height of one virus. To find the number of rows, we divide the available space by two times the height of a virus.
- We use floor division because we can only make an integer number of rows.

```
number_rows = available_height_y // (2 * virus_height)
```

ADDING ROWS

Now that we know how many rows fit in a group, we can repeat the code for creating a row

virus_invasion.py

```
def _create_group(self):
    --snip--
    virus = Virus(self)
1   virus_width, virus_height = virus.rect.size
    available_space_x = self.settings.screen_width - (2 * virus_width)
    number_viruses_x = available_space_x // (2 * virus_width)
    # Determine the number of rows of viruses that fit on the screen.
    robot_height = self.robot.rect.height
2   available_space_y = (self.settings.screen_height -(3 * virus_height) - robot_height)
    number_rows = available_space_y // (2 * virus_height)

    # Create the full group of viruses.
3   for row_number in range(number_rows):
        for virus_number in range(number_viruses_x):
            self._create_virus(virus_number, row_number)
def _create_virus(self, virus_number, row_number):
    """Create a virus and place it in the row."""
    virus = Virus(self)
    virus_width, virus_height = virus.rect.size
    virus.x = virus_width + 2 * virus_width * virus_number
    virus.rect.x = virus.x
4   virus.rect.y = virus.rect.height + 2 * virus.rect.height * row_number
    self.viruses.add(virus)
```

ADDING ROWS

1. We need the width and height of a virus, so we use the attribute size, which contains a tuple with the width and height of a rect object.

2. To calculate the number of rows we can fit on the screen, we write our available **_space_y** calculation right after the calculation for **available_space_x** .

3. To create multiple rows, we use two nested loops: one outer and one inner loop.
   - The inner loop creates the viruses in one row. The outer loop counts from 0 to the number of rows we want;
   - Python uses the code for making a single row and repeats it **number_rows** times. To nest the loops, write the new for loop and indent the code you want to repeat. Now when we call **_create_virus()**, we include an argument for the row number so each row can be placed farther down the screen.

4. The definition of _create_virus() needs a parameter to hold the row number. Within **_create_virus()**, we change a **virus's y-coordinate** value when it's not in the first row x by starting with one virus's height to create empty space at the top of the screen. Each row starts two virus heights below the previous row, so we multiply the virus height by two and then by the row number.
   - The first row number is 0, so the vertical placement of the first row is unchanged. All subsequent rows are placed farther down the screen.