PYGAME INVADERS

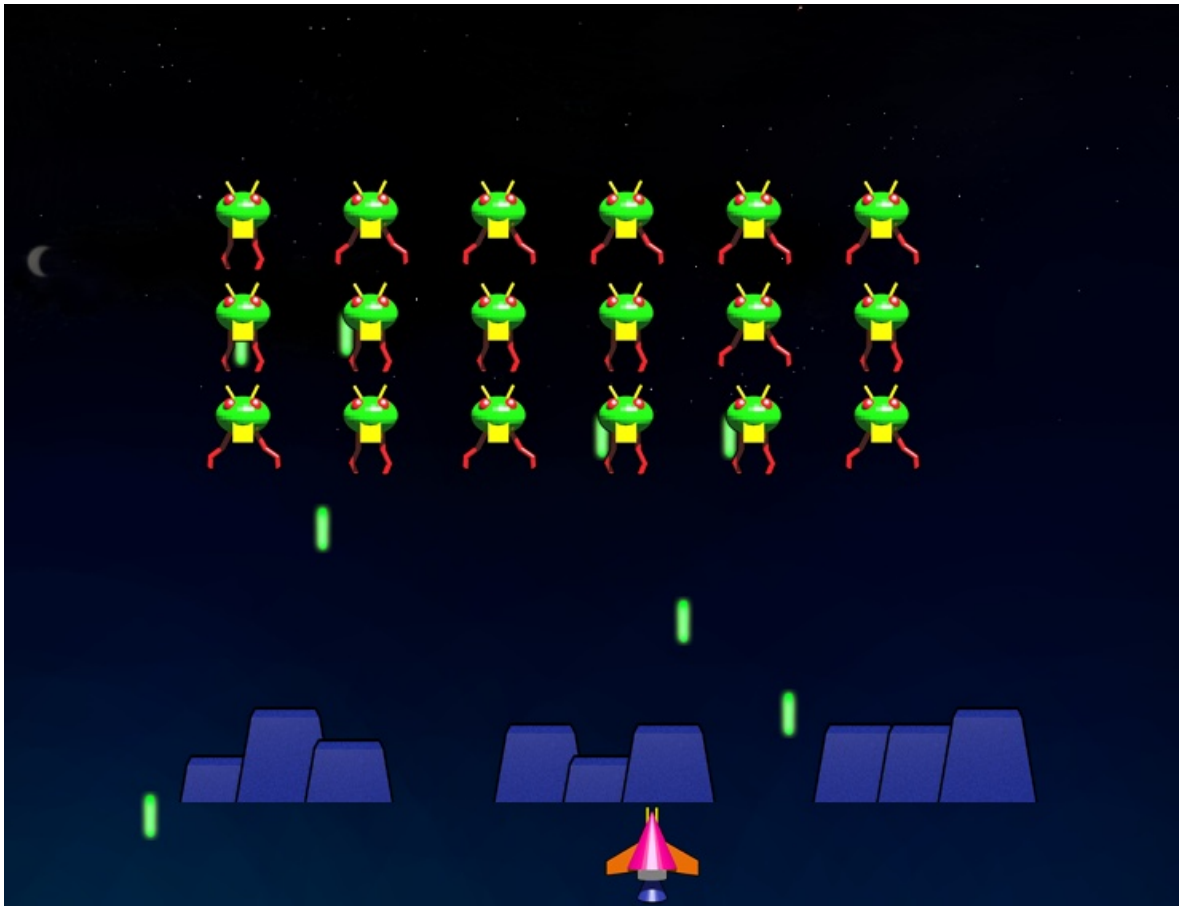
Mission Briefing

You are to create a training application that will allow potential pilots to train in defending the local bases from the attacking alien force.

The following secret documentation will provide sufficient guidance in development the application.

It has been decided that the application will, for compatibilty reasons, be created in Python 3 using the PyGame Zero framework.

Please read on for further information.



Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Workstation Requirements

Windows:

Visual Studio Code: https://code.visualstudio.com/download

Python 3: https://www.python.org/downloads/

Pygame Zero: https://pygame-zero.readthedocs.io/en/stable/installation.html

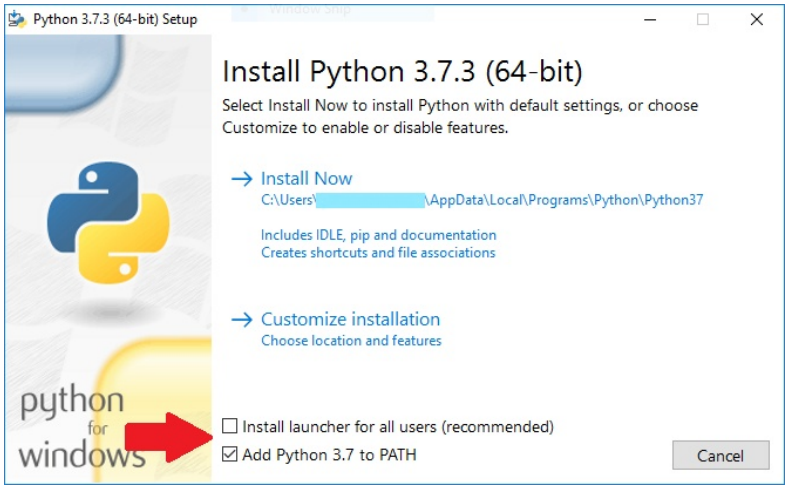
Setting up the Software:

If the software listed above has not already been installed then:

Install Python 3

Python 3 can be installed in your user account or system wide. Installing system wide may require an administrator password.

To install locally into your user account, start the installation as normal but make sure the '**Install Launcher for all users**' is unticked and '**Add Python 3.7 to PATH**' is ticked. The screen should look like this:



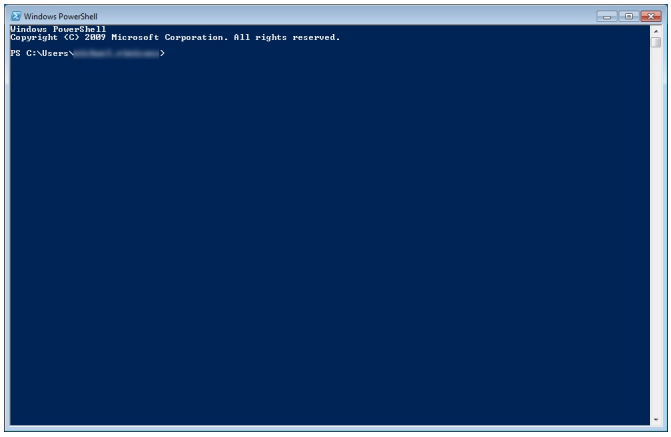
Follow the on-screen instructions to complete the installation.

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Now we have Python installed the next step is to install Pygame Zero. Again this can be installed system wide or just for the user. We will be installing for the user.

Click the start button and start PowerShell



To install for the user, type the following and press enter:

```
pip3.exe install --user pgzero
```

If required, to install system wide (this may require an administrator password), type and press enter:

```
pip3.exe install pgzero
```

This will then install the required packages for Pygame Zero.

If you encounter an error similar to:

```
SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: self
signed certificate in certificate chain
```

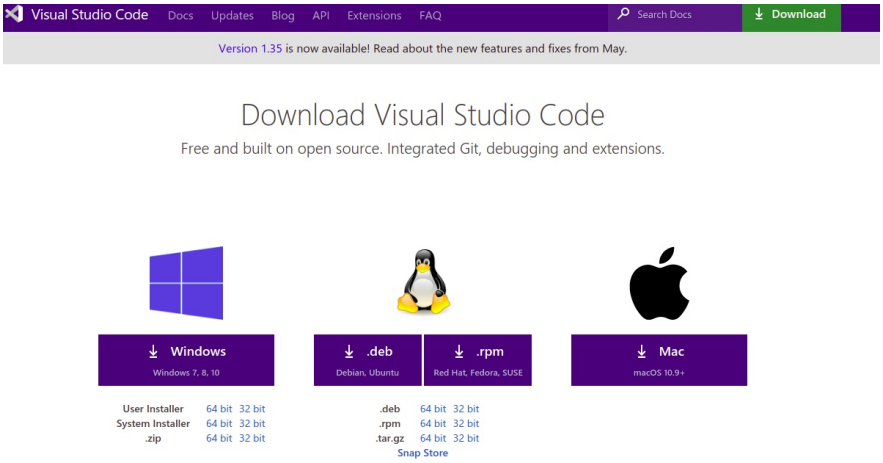
Then use the following to install PyGame Zero:

```
pip3.exe install --trusted-host pypi.org --trusted-host
files.pythonhosted.org pgzero
```

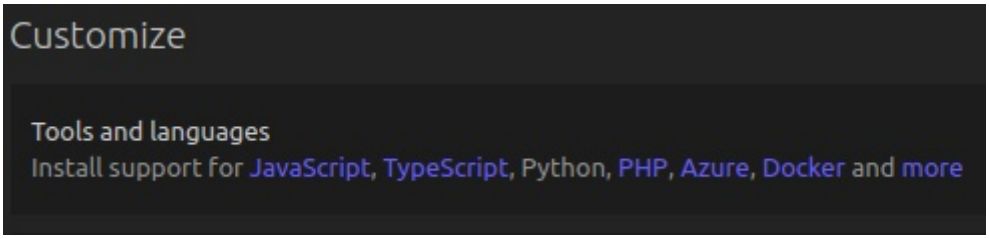
Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Download the **User Installer** for Visual Studio Code as shown below and follow the on screen prompt to install the application.



Start Visual Studio Code and install the Python extension by clicking on Python in the Customize box:



Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Once eveything has finished installing we can qucikly check to make sure everything is correct.

Start Visual Studio Code and start a new file by clicking on **File** menu and the clicking on the **New File**.

Then save this new file...any name will do but it must end with the extension.py. Use the "Save as type" menu to select python file type.

For example if you chose to name the file cats then you would call it cats.py

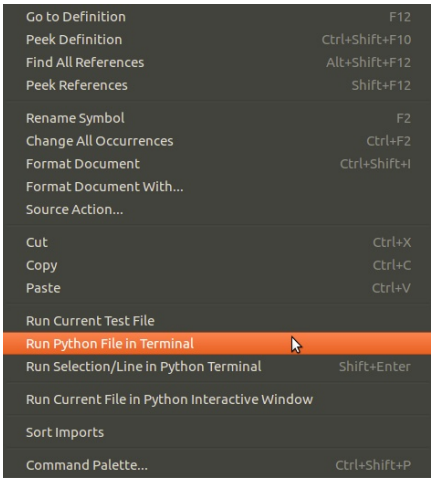
Once saved type the code below into the code window:

```
import pgzrun
WIDTH = 300
HEIGHT = 300

def draw():
    screen.fill((128, 0, 0))

pgzrun.go()
```

Save the file and then right click in the code entry screen and click on Run Python In Terminal:

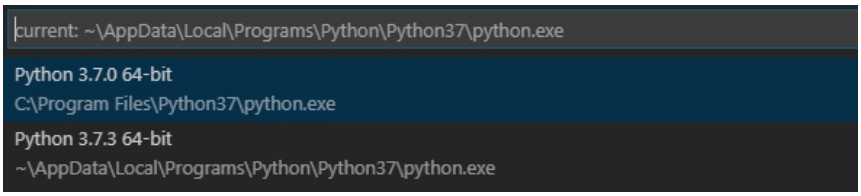


Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

If everything is correct then you should have a window with a red square on the screen and you can close this window when ready.

If an error appears stating that a Python interpreter can not be found then hold down the `ctrl` and `shift` key and press *p*. In the box that appears at the top of the screen type *Select Interpreter*.

Look in the drop down box and chose the relevant Python interpreter. An example of which can be seen below:



If more than is found then choose the one begining with *~/AppData*

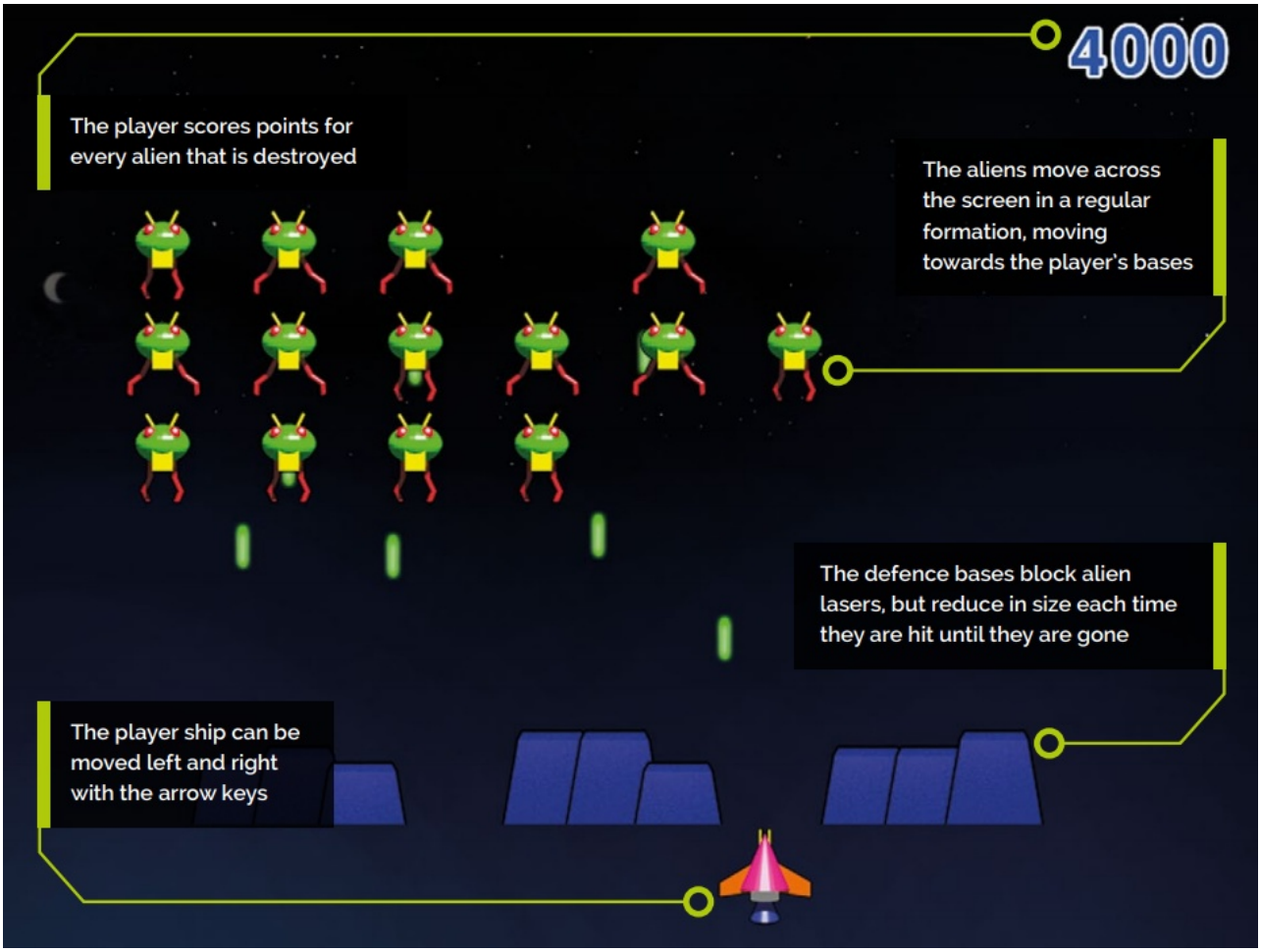
Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Creating the App

The screen mock up below shows how the application should look and also basic descriptions on how each element should operate.

Over the following pages, we will proceed step by step to create this game.

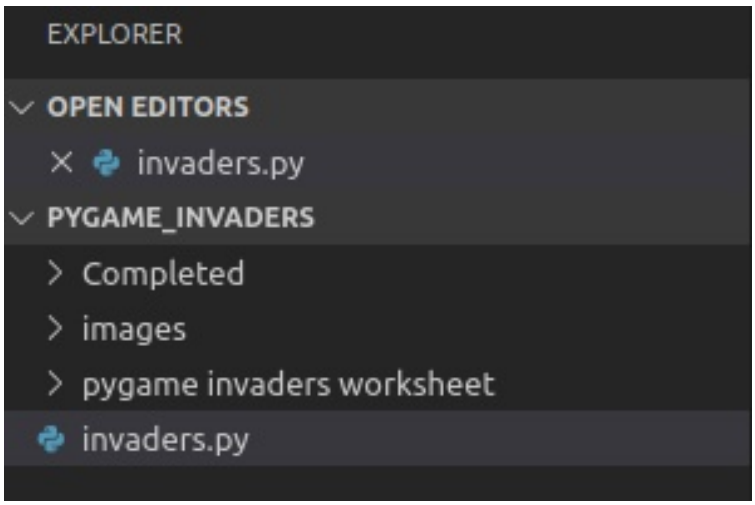


Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

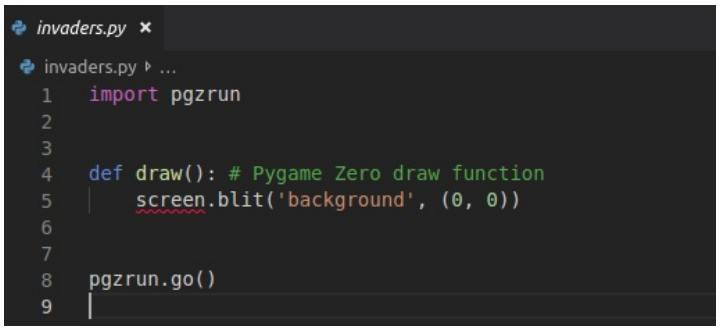
Click **Open Folder** in the **File** Menu and open the pygame invaders folder.

Notice how the explorer window on the left hand side shows the content of the folder? This allows you to keep track of the files that are used in this project.



So far there is only one file listed: invaders.py

If you click on invaders.py the contents will be displayed in the code entry box:



This code will start the PyGame Zero system and then create a window which displays the background (background.png) image from the images folder.

Right click on the code and select Run Python File in Terminal. Close the window when you have finished looking at the space scene.

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Now we have the background the next thing that will be added will be the player ship.

To do this the following code needs to be added.

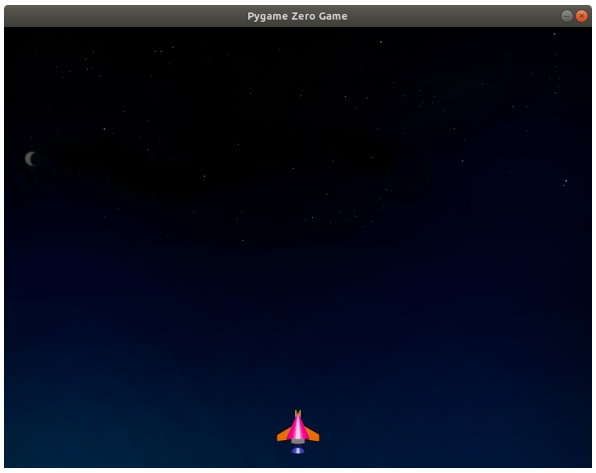
Under `import pgzrun` type the following:

```
player = Actor("player", (400, 550))
```

and in the `draw()` function, add the following under `screen.blit()` line:

```
player.draw()
```

Once done, right click and run the script like before. You should now see something like this:



and your code should look like this:

```
import pgzrun
player = Actor("player", (400, 550))

def draw():
    screen.blit('background', (0, 0))
    player.draw()

pgzrun.go()
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Now the player ship is displayed properly it's time to set up some controls to allow you to control the ship.

To enable this we need to create two new functions `update()` and `checkKeys()`.

The `update()` function allows PyGame Zero to update the screen when the game is running.

The `checkKeys()` function reads the input from the keyboard and adjusts the ships x position as required.

Type the following between the `draw()` function and `pgzrun.go()`

```
def update():
    global player
    checkKeys()

def checkKeys():
    global player
    if keyboard.left:
        if player.x > 40: player.x -= 5
    if keyboard.right:
        if player.x < 760: player.x += 5
```

As we are using Python, If you type a colon, **:**, then any lines after it need to be indented. Indented mean that the lines are started further away from the left hand margin.

Although it's possible to do this by using a single tab the preferred pythonic way is to use four spaces. However this is down to personal preference but you **can not mix** both types of indentation within your code.

If you run the code now the player ship should now move when the left and right cursor keys are pressed.

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Our ship can now move...however would it be better if it was able to defend itself?

We will now enable the laser cannon for the ship.

The following functions are needed: `makeLaserActive()`, `updateLasers()`, `drawLasers()` and `init()`. We will also need to make some additions to `draw()`, `update()` and `checkKeys()`

Update the `draw()` function to draw the laser as required:

```
def draw():
    screen.blit('background', (0, 0))
    player.draw()
    drawLasers()
```

Update the `update()` function to ensure the game knows what the lasers are doing:

```
def update():
    global player,lasers
    checkKeys()
    updateLasers()
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Update the `checkKeys()` function:

```
def checkKeys():
    global player, lasers
    if keyboard.left:
        if player.x > 40: player.x -= 5
    if keyboard.right:
        if player.x < 760: player.x += 5
    if keyboard.space:
        if player.laserActive == 1:
            player.laserActive = 0
            clock.schedule(makeLaserActive, 1.0)
            l = len(lasers)
            lasers.append(Actor("laser2",(player.x,player.y-32)))
            lasers[l].status = 0
            lasers[l].type = 1
```

Add the `makeLaserActive()` function after the `checkKeys()` function:

```
def makeLaserActive():
    global player
    player.laserActive = 1
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

Add `updateLasers()` after the `makeLaserActive()` function:

```
def updateLasers():
    global lasers
    for l in range(len(lasers)):
        if lasers[l].type == 1:
            lasers[l].y -= 5
            if lasers[l].y < 10: lasers[l].status = 1
```

Insert the `drawLasers()` function between the the `update()` and `checkKeys()` functions:

```
def drawLasers():
    for l in range(len(lasers)): lasers[l].draw()
```

Add the `init()` function between `updateLasers()` and the `pgzrun.go()` on the last line:

```
def init():
    global lasers
    lasers = []
    player.laserActive = 1
```

Finally add `init()` one line above `pgzrun.go()`

Run the code and your ship should now be able to fire lasers.

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Your code should look something like the following:

```
import pgzrun
player = Actor("player",(400,550))

def draw():
     screen.blit('background',(0,0))
     player.draw()
     drawLasers()

def update():
     global player
     checkKeys()
     updateLasers()

def drawLasers():
     for l in range(len(lasers)): lasers[l].draw()

def checkKeys():
     global player, lasers
     if keyboard.left:
          if player.x > 40: player.x -= 5
     if keyboard.right:
          if player.x < 760: player.x += 5
     if keyboard.space:
          if player.laserActive == 1:
               player.laserActive = 0
               clock.schedule(makeLaserActive,1.0)
               l = len(lasers)
               lasers.append(Actor("laser2",(player.x,player.y-32)))
               lasers[l].status = 0
               lasers[l].type = 1

def makeLaserActive():
     global player
     player.laserActive = 1
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

```
def updateLasers():
    global lasers
    for l in range(len(lasers)):
        if lasers[l].type == 1:
            lasers[l].y -= 5
            if lasers[l].y < 10: lasers[l].status = 1

def init():
    global lasers
    lasers = []
    player.laserActive = 1

init()
pgzrun.go()
```

Now it's time to add some bases to help the ship.

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

To add the bases we need to add the new functions `drawClipped()`, `initBases()`, `drawBases()` and `checkBases()`. These will keep control of the bases.
The `draw()` and `init()` functions will also need to be updated.

Update the `draw()` function as below:

```
def draw():
    screen.blit('background', (0, 0))
    player.draw()
    drawLasers()
    drawBases()
```

Create the `drawBases()` function in between the `update()` and `init()` functions:

```
def drawBases():
    for b in range(len(bases)): bases[b].drawClipped()
```

Next create the `checkBases()` function between `makeLaserActive()` and `updateLasers()`:

```
def checkBases():
    for b in range(len(bases)):
        if l < len(bases):
            if bases[b].height < 5:
                del bases[b]
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

`drawClipped()` and `initBases()` need to be created after the `init()` function:

```
def drawClipped(self):
    screen.surface.blit(self._surf, (self.x-32, self.y-
self.height+30),(0,0,64,self.height))

def initBases():
    global bases
    bases = []
    bc = 0
    for b in range(3):
        for p in range(3):
            bases.append(Actor("base1", midbottom=(150+(b*200)+(p*40),520)))
            bases[bc].drawClipped = drawClipped.__get__(bases[bc])
            bases[bc].height = 60
            bc +=1
```

Finally update the `init()` function:

```
def init():
    global lasers
    lasers = []
    player.laserActive = 1
    initBases()
```

Run the script and there should now be three bases above the space ship.

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Your code should now look something like this:

```
import pgzrun
player = Actor("player",(400,550))

def draw():
     screen.blit('background',(0,0))
     player.draw()
     drawLasers()
     drawBases()

def update():
     global player
     checkKeys()
     updateLasers()

def drawBases():
     for b in range(len(bases)): bases[b].drawClipped()

def drawLasers():
     for l in range(len(lasers)): lasers[l].draw()

def checkKeys():
     global player, lasers
     if keyboard.left:
          if player.x > 40: player.x -= 5
     if keyboard.right:
          if player.x < 760: player.x += 5
     if keyboard.space:
          if player.laserActive == 1:
               player.laserActive = 0
               clock.schedule(makeLaserActive,1.0)
               l = len(lasers)
               lasers.append(Actor("laser2",(player.x,player.y-32)))
               lasers[l].status = 0
               lasers[l].type = 1
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

```
def makeLaserActive():
     global player
     player.laserActive = 1

def updateLasers():
     global lasers
     for l in range(len(lasers)):
          if lasers[l].type == 1:
               lasers[l].y -= 5
               if lasers[l].y < 10: lasers[l].status = 1

def checkBases():
     for b in range(len(bases)):
          if l < len(bases):
               if bases[b].height < 5:
                    del bases[b]

def init():
     global lasers
     lasers = []
     player.laserActive = 1
     initBases()

def drawClipped(self):
     screen.surface.blit(self._surf,(self.x-32,self.y-
self.height+30),(0,0,64,self.height))

def initBases():
     global bases
     bases = []
     bc = 0
     for b in range(3):
          for p in range(3):
     bases.append(Actor("base1",midbottom=(150+(b*200)+(p*40),520)))
               bases[bc].drawClipped = drawClipped.__get__(bases[bc])
               bases[bc].height = 60
               bc += 1
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

```
init()
pgzrun.go()
```

With all that now done and working it's time to add the aliens.

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

To add the aliens into the app we need to create three functions: `drawAliens()`, `updateAliens()`, `initAliens()` and to update the following functions: `draw()` and `init()`.

The `drawAliens()` below function is needs to be placed the `drawBases()` function.

```
def drawAliens():
    for a in range(len(aliens)): aliens[a].draw()
```

`updateAliens()` is added after `drawAliens()`:

```
def updateAliens():
    global moveSequence, lasers, moveDelay
    movex = movey = 0
    for a in range(len(aliens)):
        animate(aliens[a], pos=(aliens[a].x + movex, aliens[a].y + movey),
duration=0.5, tween='linear')
        if randint(0, 1) == 0:
            aliens[a].image = "alien1"
        else:
            aliens[a].image = "alien1b"
        if aliens[a].y > player.y and player.status == 0:
            player.status = 1
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

`initAliens()` is added after the `init()` function:

```
def initAliens():
    global aliens
    aliens = []
    for a in range(18):
        aliens.append(Actor("alien1", (210+(a % 6)*80,100+(int(a/6)*64))))
        aliens[a].status = 0
```

The `draw()` and `init()` and functions also need to be updated to place the aliens on the screen

```
def draw():
    screen.blit('background', (0, 0))
    player.draw()
    drawLasers()
    drawAliens()


def init():
    global lasers,moveSequence,moveDelay
    lasers = []
    player.laserActive = 1
    initBases()
    initAliens()
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

Run the script and you should notice an alien armada has appeared.

However, there appears to be an issue with the aliens...they appear to be stationary!

To make it more of a challenge, and to give the aliens a chance, let's create some code to move the aliens left and right on the screen.

To start with, as we will be making use of random numbers we need to import `randint` from the `random` library. To do this add the following line under import pgzrun at the top of the code:

```
from random import randint()
```

A variable, named `DIFFICULTY`, also needs to be defined and set to 1. This should be entered on the line after the randint() line above to look like the following.

```
from random import randint()
DIFFICULTY = 1
```

The `update()`, `updateAliens()`, `updateLasers()` and `init()` functions needs to be updated as below to allow the aliens to move on each update:

```
def update():
    global moveCounter,player
    if player.status < 30 and len(aliens) > 0:
        checkKeys()
        updateLasers()
        moveCounter += 1
        if moveCounter == moveDelay:
            moveCounter = 0
            updateAliens()
        if player.status > 0: player.status += 1
    else:
        if keyboard.RETURN: init()
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

```
def init():
    global lasers,moveSequence,moveDelay, moveCounter
    lasers = []
    moveCounter = moveSequence = player.status = score =
player.laserCountdown = 0
    player.laserActive = 1
    moveDelay = 30
    initBases()
    initAliens()
```

```
def updateAliens():
    global moveSequence, lasers, moveDelay
    movex = movey = 0
    if moveSequence < 10 or moveSequence > 30: movex =  -15
     if moveSequence == 10 or moveSequence == 30:
          movey = 50 + (10*DIFFICULTY)
          moveDelay -= 1
     if moveSequence > 10 and moveSequence < 30: movex = 15
    for a in range(len(aliens)):
        animate(aliens[a], pos=(aliens[a].x + movex, aliens[a].y + movey),
duration=0.5, tween='linear')
        if randint(0, 1) == 0:
            aliens[a].image = "alien1"
        else:
            aliens[a].image = "alien1b"
            if randint(0,5) == 0:
                    lasers.append(Actor("laser1",(aliens[a].x,aliens[a].y)))
                    lasers[len(lasers)-1].status = 0
                    lasers[len(lasers)-1].type = 0
        if aliens[a].y > player.y and player.status == 0:
            player.status = 1
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

```
def updateLasers():
     global lasers
     for l in range(len(lasers)):
          if lasers[l].type == 0:
               lasers[l].y += (2*DIFFICULTY)
               if lasers[l].y > 600: lasers[l].status = 1
          if lasers[l].type == 1:
               lasers[l].y -= 5
               if lasers[l].y < 10: lasers[l].status = 1
```

Run the script now and the alien armada will be moving left to right across the screen and firing lasers at the ship. Don't worry as the alien lasers are unable to damage the ship for now...

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Your code should look like this now:

```
import pgzrun
player = Actor("player",(400,550))
from random import randint
DIFFICULTY = 1

def draw():
     screen.blit('background',(0,0))
     player.draw()
     drawLasers()
     drawBases()
     drawAliens()

def update():
     global moveCounter,player
     if player.status < 30 and len(aliens) > 0:
          checkKeys()
          updateLasers()
          moveCounter += 1
          if moveCounter == moveDelay:
               moveCounter = 0
               updateAliens()
          if player.status > 0: player.status += 1
     else:
          if keyboard.Return: init()

def drawBases():
     for b in range(len(bases)): bases[b].drawClipped()

def drawLasers():
     for l in range(len(lasers)): lasers[l].draw()

def drawAliens():
     for a in range(len(aliens)): aliens[a].draw()
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

```
def updateAliens():
     global moveSequence,lasers,moveDelay
     movex = movey = 0
     if moveSequence < 10 or moveSequence > 30: movex =  -15
     if moveSequence == 10 or moveSequence == 30:
          movey = 50 + (10*DIFFICULTY)
          moveDelay -= 1
     if moveSequence > 10 and moveSequence < 30: movex = 15
     for a in range(len(aliens)):
          animate(aliens[a], pos=(aliens[a].x+movex,
aliens[a].y+movey),duration=0.5,tween='linear')
          if randint(0,1) == 0:
               aliens[a].image = "alien1"
          else:
               aliens[a].image = "alien1b"
               if randint(0,5) == 0:
          lasers.append(Actor("laser1",(aliens[a].x,aliens[a].y)))
                    lasers[len(lasers)-1].status = 0
                    lasers[len(lasers)-1].type = 0
          if aliens[a].y > player.y and player.status == 0:
               player.status = 1
     moveSequence += 1
     if moveSequence == 40: moveSequence = 0

def checkKeys():
     global player, lasers
     if keyboard.left:
          if player.x > 40: player.x -= 5
     if keyboard.right:
          if player.x < 760: player.x += 5
     if keyboard.space:
          if player.laserActive == 1:
               player.laserActive = 0
               clock.schedule(makeLaserActive,1.0)
               l = len(lasers)
               lasers.append(Actor("laser2",(player.x,player.y-
32)))
               lasers[l].status = 0
               lasers[l].type = 1
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

```
def makeLaserActive():
     global player
     player.laserActive = 1

def updateLasers():
     global lasers
     for l in range(len(lasers)):
          if lasers[l].type == 0:
               lasers[l].y += (2*DIFFICULTY)
               if lasers[l].y > 600: lasers[l].status = 1
          if lasers[l].type == 1:
               lasers[l].y -= 5
               if lasers[l].y < 10: lasers[l].status = 1

def checkBases():
     for b in range(len(bases)):
          if l < len(bases):
               if bases[b].height < 5:
                    del bases[b]

def init():
     global lasers, moveSequence, moveDelay, moveCounter
     lasers = []
     moveCounter = moveSequence = player.status = score =
player.laserCountdown = 0
     player.laserActive = 1
     moveDelay = 30
     initBases()
     initAliens()

def initAliens():
     global aliens
     aliens = []
     for a in range(18):

aliens.append(Actor("alien1",(210+(a%6)*80,100+(int(a/6)*64))))
          aliens[a].status = 0
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

```
def drawClipped(self):
     screen.surface.blit(self._surf,(self.x-32,self.y-
self.height+30),(0,0,64,self.height))

def initBases():
     global bases
     bases = []
     bc = 0
     for b in range(3):
          for p in range(3):

bases.append(Actor("base1",midbottom=(150+(b*200)+(p*40),520)))
               bases[bc].drawClipped =
drawClipped.__get__(bases[bc])
               bases[bc].height = 60
               bc += 1
init()
pgzrun.go()
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

So far we have our ship, bases and aliens on the screen but nothing happens when either the ship lasers or the alien lasers hit anything. Lets get this fixed.

First let's create the code to cause damage to the bases when they are hit by laser fire.

Two of the functions, `updateLasers()` and `initBases()` need slight additions as below:

```
def updateLasers():
    global lasers, aliens
    for l in range(len(lasers)):
        if lasers[l].type == 0:
            lasers[l].y += (2*DIFFICULTY)
            checkLaserHit(l)
            if lasers[l].y > 600: lasers[l].status = 1
        if lasers[l].type == 1:
            lasers[l].y -= 5
            checkPlayerLaserHit(l)
            if lasers[l].y < 10: lasers[l].status = 1
    lasers = listCleanup(lasers)
    aliens = listCleanup(aliens)
```

```
def initBases():
    global bases
    bases = []
    bc = 0
    for b in range(3):
        for p in range(3):
            bases.append(Actor("base1", midbottom=(150+(b*200)+(p*40),520)))
            bases[bc].drawClipped = drawClipped.__get__(bases[bc])
            bases[bc].collideLaser = collideLaser.__get__(bases[bc])
            bases[bc].height = 60
            bc +=1
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

We then need to create the following function to help our code keep track of what's been hit. Create it between the `updateLasers()` and `checkBases()` functions:

```
def listCleanup(l):
    newList = []
    for i in range(len(l)):
        if l[i].status == 0: newList.append(l[i])
    return newList
```

Next we create the following functions to check for any laser collisions.

Place the following two functions after `listCleanup()` function:

```
def checkLaserHit(l):
    global player
    if player.collidepoint((lasers[l].x, lasers[l].y)):
        player.status = 1
        lasers[l].status = 1
    for b in range(len(bases)):
        if bases[b].collideLaser(lasers[l]):
            bases[b].height -= 10
            lasers[l].status = 1

def checkPlayerLaserHit(l):
    for b in range(len(bases)):
        if bases[b].collideLaser(lasers[l]): lasers[l].status = 1
    for a in range(len(aliens)):
        if aliens[a].collidepoint((lasers[l].x, lasers[l].y)):
            lasers[l].status = 1
            aliens[a].status = 1
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

Now we can add the code that works out if there has been any collisions.

```
def collideLaser(self, other):
    return (
        self.x-20 < other.x+5 and
        self.y-self.height+30 < other.y and
        self.x+32 > other.x+5 and
        self.y-self.height+30 + self.height > other.y
    )
```

If you run the code now you will notice that the bases will fall further apart each time they are hit by the alien lasers and each time an alien is hit it will be destroyed. However if your ship is hit the game will freeze. Let's see if we can fix this by making the ship explode.

First we need to import another module, math, so at the top of the code add the following line under the `from random import randint`

```
import math
```

Next the draw() functoin needs to be updated:

```
def draw(): # Pygame Zero draw function
    screen.blit('background', (0, 0))
    player.image = player.images[math.floor(player.status/6)]
    player.draw()
    drawLasers()
    drawBases()
    drawAliens()
```

Watch out for the square brackets in the new line.

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Update the `init()` next to make our code aware of the different images for the ship.

```
def init():
    global lasers,moveSequence,moveDelay, moveCounter
    lasers = []
    moveCounter = moveSequence = player.status = score =
player.laserCountdown = 0
    player.laserActive = 1
    moveDelay = 30
    player.images =
["player","explosion1","explosion2","explosion3","explosion4","explosion5"]
    initBases()
    initAliens()
```

Run this code and let the ship be hit by a alien laser. It will now explode apart but the game still freezes. This is becauses once our ship has been destroyed PyGame Zero has nothing for you to control and stops the game. If you press the enter key on your keyboard the game will restart.

As this is not obvious to the player lets display a message on the screen informing them to press enter.

Update the `draw()` function:

```
def draw(): # Pygame Zero draw function
    screen.blit('background', (0, 0))
    player.image = player.images[math.floor(player.status/6)]
    player.draw()
    drawLasers()
    drawBases()
    drawAliens()
    if player.status >= 30:
        screen.draw.text("GAME OVER\nPress Enter to play again" ,
center=(400, 300), owidth=0.5, ocolor=(255,255,255), color=(255,64,0) ,
fontsize=60)
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

Run the code and let the ship be hit by a laser. After the explosion the game over message will be displayed.

But what happens if we win and all the aliens are destroyed?

Exactly the same as what happend before if the ship was destroyed and before we created the code to display the Game Over.

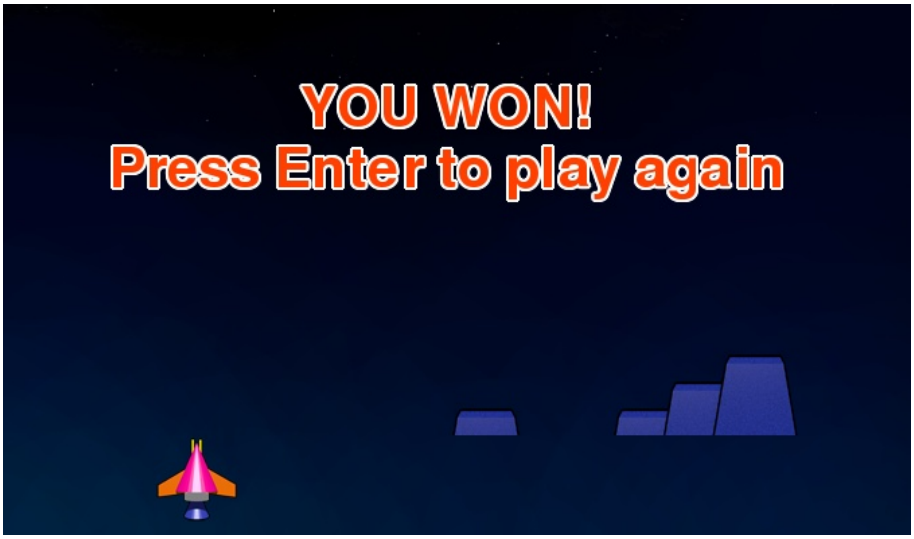
This time we need to create some code to display a message when you win. Enter the following line after the the line we entered in `draw()` for the Game Over message:

```
if len(aliens) == 0 :
        screen.draw.text("YOU WON!\nPress Enter to play again" , center=(400,
300), owidth=0.5, ocolor=(255,255,255), color=(255,64,0) , fontsize=60)
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

Now, when you destroy all the aliens, a "You Won" message is displayed.



Finally, to finish it off, we need to add some code to display a score and update it each time an alien is hit.

Update the checkPlayerLaserHit() function:

```
def checkPlayerLaserHit(l):
    global score
    for b in range(len(bases)):
        if bases[b].collideLaser(lasers[l]):
lasers[l].status = 1
    for a in range(len(aliens)):
        if aliens[a].collidepoint((lasers[l].x,
lasers[l].y)):
            lasers[l].status = 1
            aliens[a].status = 1
            score += 1000
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

Update the second line of the `init()` function:

```
def init():
    global lasers,score,moveSequence,moveDelay, moveCounter
```

Update the `draw()` function with the following line between the `drawAliens()` and `If player.status >= 30` lines:

```
screen.draw.text(str(score) , topright=(780, 10), owidth=0.5,
ocolor=(255,255,255), color=(0,64,255) , fontsize=60)
```

Run the code and there will now be a score counter in the top right hand corner of the screen.

Your code should look similar to the following:

PYGAME INVADERS

```
import pgzrun
from random import randint
import math

player = Actor("player",(400,550))
DIFFICULTY = 1

def draw():
     screen.blit('background',(0,0))
     player.image = player.images[math.floor(player.status/6)]
     player.draw()
     drawLasers()
     drawBases()
     drawAliens()
     screen.draw.text(str(score), topright=(780,10), owidth=0.5,
ocolor=(255,255,255), color=(0,64,255),fontsize = 60)
     if player.status >= 30:
          screen.draw.text("GAME OVER!!\nPress Enter to try
again",center=(400,300), owidth=0.5, ocolor=(255,255,255),
color=(255,64,0),fontsize = 60)
     if len(aliens) == 0:
          screen.draw.text("YOU WON!!\nPress Enter to try
again",center=(400,300), owidth=0.5, ocolor=(255,255,255),
color=(255,64,0),fontsize = 60)

def update():
     global moveCounter,player
     if player.status < 30 and len(aliens) > 0:
          checkKeys()
          updateLasers()
          moveCounter += 1
          if moveCounter == moveDelay:
               moveCounter = 0
               updateAliens()
          if player.status > 0: player.status += 1
     else:
          if keyboard.RETURN: init()
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

```
def drawBases():
     for b in range(len(bases)): bases[b].drawClipped()

def drawLasers():
     for l in range(len(lasers)): lasers[l].draw()

def drawAliens():
     for a in range(len(aliens)): aliens[a].draw()

def updateAliens():
     global moveSequence,lasers,moveDelay
     movex = movey = 0
     if moveSequence < 10 or moveSequence > 30: movex =  -15
     if moveSequence == 10 or moveSequence == 30:
          movey = 50 + (10*DIFFICULTY)
          moveDelay -= 1
     if moveSequence > 10 and moveSequence < 30: movex = 15
     for a in range(len(aliens)):
          animate(aliens[a], pos=(aliens[a].x+movex,
aliens[a].y+movey),duration=0.5,tween='linear')
          if randint(0,1) == 0:
               aliens[a].image = "alien1"
          else:
               aliens[a].image = "alien1b"
               if randint(0,5) == 0:
                    lasers.append(Actor("laser1",(aliens[a].x,aliens[a].y)))
                    lasers[len(lasers)-1].status = 0
                    lasers[len(lasers)-1].type = 0
          if aliens[a].y > player.y and player.status == 0:
               player.status = 1
     moveSequence += 1
     if moveSequence == 40: moveSequence = 0
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

```
def checkKeys():
     global player, lasers
     if keyboard.left:
          if player.x > 40: player.x -= 5
     if keyboard.right:
          if player.x < 760: player.x += 5
     if keyboard.space:
          if player.laserActive == 1:
               player.laserActive = 0
               clock.schedule(makeLaserActive,1.0)
               l = len(lasers)
               lasers.append(Actor("laser2",(player.x,player.y-32)))
               lasers[l].status = 0
               lasers[l].type = 1

def makeLaserActive():
     global player
     player.laserActive = 1

def updateLasers():
     global lasers,aliens
     for l in range(len(lasers)):
          if lasers[l].type == 0:
               lasers[l].y += (2*DIFFICULTY)
               checkLaserHit(l)
               if lasers[l].y > 600: lasers[l].status = 1
          if lasers[l].type == 1:
               lasers[l].y -= 5
               checkPlayerLaserHit(l)
               if lasers[l].y < 10: lasers[l].status = 1
     lasers = listCleanup(lasers)
     aliens = listCleanup(aliens)

def listCleanup(l):
     newList = []
     for i in range(len(l)):
          if l[i].status == 0 : newList.append(l[i])
     return newList
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

```
def checkLaserHit(l):
     global player
     if player.collidepoint(lasers[l].x, lasers[l].y):
          player.status = 1
          lasers[l].status = 1
     for b in range(len(bases)):
          if bases[b].collideLaser(lasers[l]):
               bases[b].height -= 10
               lasers[l].status = 1

def checkPlayerLaserHit(l):
     global score
     for b in range(len(bases)):
          if bases[b].collideLaser(lasers[l]): lasers[l].status = 1
     for a in range(len(aliens)):
          if aliens[a].collidepoint((lasers[l].x, lasers[l].y)):
               lasers[l].status = 1
               aliens[a].status = 1
               score += 1000

def collideLaser(self, other):
     return(
          self.x-20 < other.x+5 and
          self.y-self.height+30 < other.y and
          self.x+32 > other.x+5 and
          self.y-self.height+30 + self.height > other.y
     )

def checkBases():
     for b in range(len(bases)):
          if l < len(bases):
               if bases[b].height < 5:
                    del bases[b]
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf

PYGAME INVADERS

```
def init():
     global lasers,score, moveSequence, moveDelay, moveCounter
     lasers = []
     moveCounter = moveSequence = player.status = score =
player.laserCountdown = 0
     player.laserActive = 1
     moveDelay = 30
     player.images =
["player","explosion1","explosion2","explosion3","explosion4","explosion5"]
     initBases()
     initAliens()

def initAliens():
     global aliens
     aliens = []
     for a in range(18):
          aliens.append(Actor("alien1",(210+(a%6)*80,100+(int(a/6)*64))))
          aliens[a].status = 0

def drawClipped(self):
     screen.surface.blit(self._surf,(self.x-32,self.y-
self.height+30),(0,0,64,self.height))

def initBases():
     global bases
     bases = []
     bc = 0
     for b in range(3):
          for p in range(3):

bases.append(Actor("base1",midbottom=(150+(b*200)+(p*40),520)))
               bases[bc].drawClipped = drawClipped.__get__(bases[bc])
               bases[bc].collideLaser = collideLaser.__get__(bases[bc])
               bases[bc].height = 60
               bc += 1
init()
pgzrun.go()
```

Based on Pygame Zero Invaders published in issue 74 of the MagPi Magazine:
https://www.raspberrypi.org/magpi-issues/MagPi74.pdf