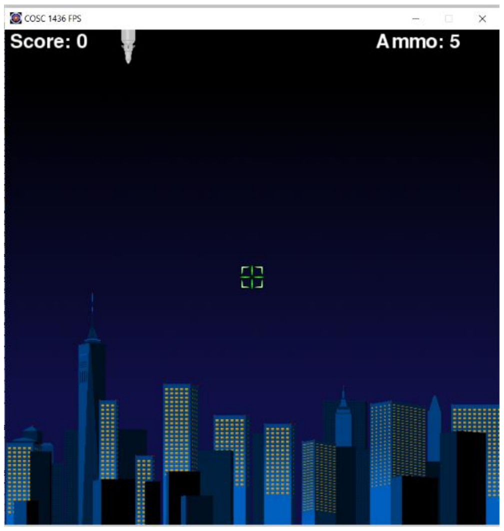
Programming Fundamentals I
Lab 10 - Functions

In this lab we are going to apply our knowledge of functions by defining and calling new functions. Here are the new additions to the game that we are going to program in this lab:

- Make the missile move from the top to the bottom of the screen
- Make the missile explode when it reaches the bottom of the screen
- Generate a new target at a random location one second after the old target explodes
- Randomly choose between two types of targets when creating a new one

EXERCISE:

1. We added a city background to our game. Identify the instruction that displays the city background on the screen.
2. Modify the position of the missile so it is positioned at the top of the screen, as shown below:

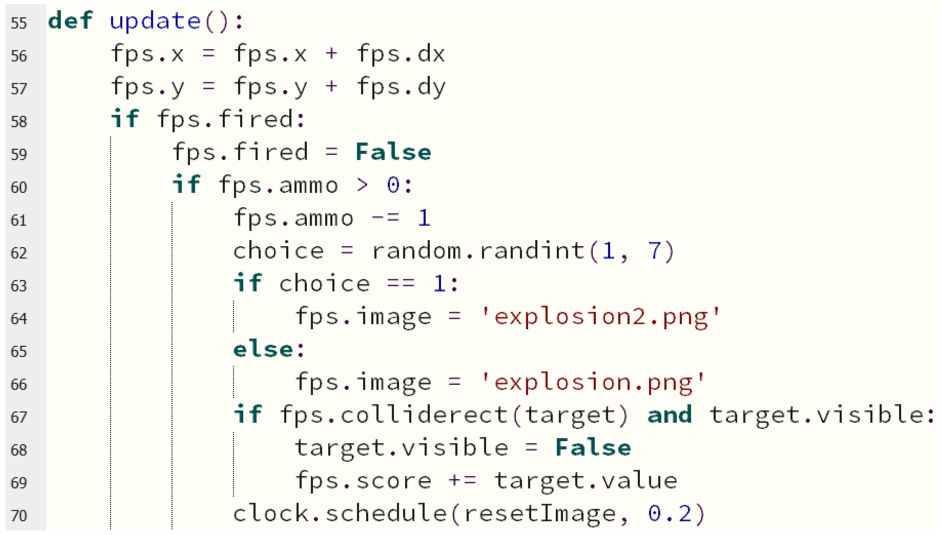


WHY DO WE NEED FUNCTIONS:

Functions are important in programming for two main reasons:

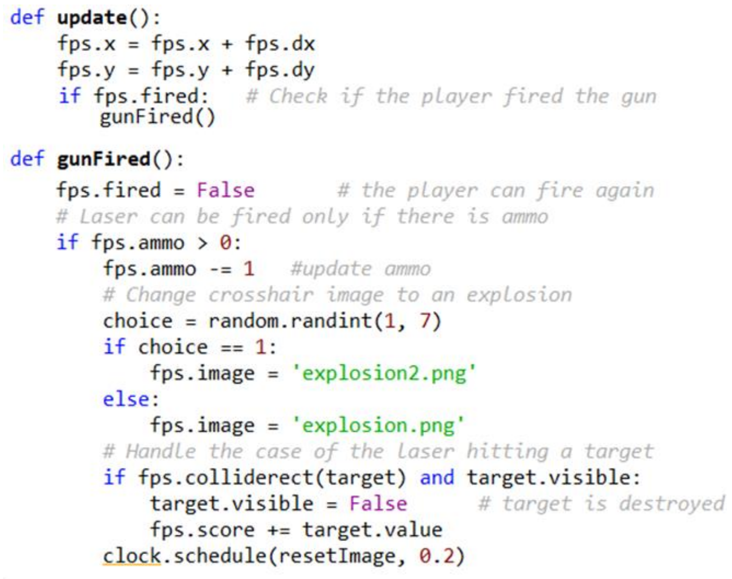
- Organize our code so it is easier to understand
- Prevent the duplication of the same code

Let us look at the `update()` function:



```
55 def update():
56     fps.x = fps.x + fps.dx
57     fps.y = fps.y + fps.dy
58     if fps.fired:
59         fps.fired = False
60         if fps.ammo > 0:
61             fps.ammo -= 1
62             choice = random.randint(1, 7)
63             if choice == 1:
64                 fps.image = 'explosion2.png'
65             else:
66                 fps.image = 'explosion.png'
67             if fps.colliderect(target) and target.visible:
68                 target.visible = False
69                 fps.score += target.value
70             clock.schedule(resetImage, 0.2)
```

Although we wrote this code ourselves, it is very easy to forget what the code is doing. Here, the code is doing two main things: 1) update the position of the crosshair, 2) check and handle the firing of the laser gun. To improve the readability of this code, we will create a new function and use comments. Let us define a new function called `gunFired()`, and move instructions 59 to 70 to this new function. Let us also add helpful comments to this function. After we are done, the code should look like this:



```
def update():
    fps.x = fps.x + fps.dx
    fps.y = fps.y + fps.dy
    if fps.fired:    # Check if the player fired the gun
       gunFired()

def gunFired():
    fps.fired = False        # the player can fire again
    # Laser can be fired only if there is ammo
    if fps.ammo > 0:
        fps.ammo -= 1    #update ammo
        # Change crosshair image to an explosion
        choice = random.randint(1, 7)
        if choice == 1:
            fps.image = 'explosion2.png'
        else:
            fps.image = 'explosion.png'
        # Handle the case of the laser hitting a target
        if fps.colliderect(target) and target.visible:
            target.visible = False      # target is destroyed
            fps.score += target.value
        clock.schedule(resetImage, 0.2)
```

EXERCISE:

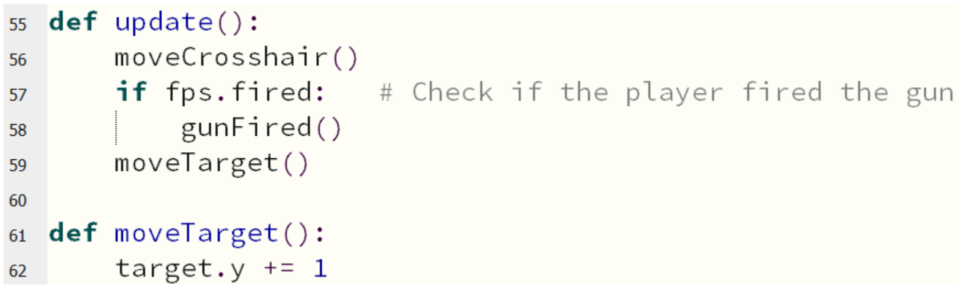
Let us improve the `update()` function a little bit more. Create a function called `moveCrosshair()`, and move liens 56 and 57 into that function. Let `update()` call the new function.

MISSILE MOVEMENT:

The next thing to do is to make the missile move from the top to the bottom of the screen. This task is not as difficult as it seems. The most important part is to divide the task into smaller parts so it is easier to solve:

1. identify the function/s that handles this feature.
2. which *Actor*, and which of its properties we need to modify?
3. what is the condition needed to stop moving the missile?

Changing the position of an *Actor* falls under game calculation, which the `update()` function handles. Therefore, to move the missile downwards we need to incrementally change its position in the `update()` function. Since the missile is only moving downwards then only the *y* position should be modified:



```
def update():
    moveCrosshair()
    if fps.fired:    # Check if the player fired the gun
        gunFired()
    moveTarget()

def moveTarget():
    target.y += 1
```

What happens when the missile reaches the end of the screen?

MISSILE EXPLODING:

We need to modify our game so the missile explodes when it reaches the bottom of the screen. We also want to penalize the player by reducing the score since the missile was not shot before it reached the city. Here are the tasks that we need to figure out:

1. what is the condition that determines the missile reached the bottom of the screen?
2. what are the steps (instructions) needed to make the missile explode? We did something similar when the player shot the missile.

EXERCISE:

1. Complete the following code snippet below:

```
def moveTarget():
    (A)---- += 1
    if -(B)----  == HEIGHT - BLOCK_SIZE:
        ----(C)--- = 'explosion3.png'
        clock.schedule(----(D)---, 0.2)


def hideTarget():
    target.visible = False      # target is destroyed
```

2. Modify the `moveTarget()` function so it reduces the score by 10 points when the missile hits the city.

CREATING A NEW MISSILE:

We need our game to create a new missile after it gets hit by a laser or explodes when it reaches the city. Although we are using the term destroyed, all that we are doing in our code is setting the *visible* flag to *False* so the `draw()` function does not display the missile on the screen.

EXERCISE:

1. Let us create a new function called `newTarget()` that will do that. This means, all we have to do in order to create a new missile is to reset the properties (variables) of the *target* Actor, and then make it visible.

All that is left to do is to call this function after the missile is destroyed when hit by a laser or exploded when it reaches the city. This means we need to call this function in two different sections of our program. This is why we declared this new function in order to prevent duplication of the code. We are going to use `clock.schedule(newTarget, 1)` to call the function after the program waits for 1 second.

EXERCISE:

1. Insert `clock.schedule(newTarget, 1)` where it is needed (2 locations).
2. We want our program to randomly choose between two types of targets: *missile1* (with a value of 5) or *missile2* (with a value of 10) when creating a new missile. Modify the `newTarget()` function to have this feature. Hint: you need to use a random number.
3. Modify the `newTarget()` function to initialize the x position to a random location.

WHAT TO HAND IN:

Submit your project electronically through D2L by attaching and submitting your **Python program file**.