Cat and Mouse Programming Tutorial

Waverly Roeger
waverlyroeger@gmail.com

Derek Jones
derekjones@asu.edu

Introduction to Programming Concepts

Variables

Print Statements

Logical Operators

If Statements

While Loops

Let's start coding!

Choose a Starting Location for the Cat

First we are going to create 2 variables named "catX" and "catY" to store the location of the cat. In this example, we set the cat's initial position to (0,0).

```
catX = 0                                                    <--
catY = 0                                                    <--
```

Hide the Mouse

Next, we want to hide the mouse somewhere - and in a similar way to the cat this is done with 2 variables, "mouseX" and "mouseY". You can choose any 2 numbers for the mouse but for this example we will use (3,3). Our code shall now look like this:

```
catX = 0
catY = 0

mouseX = 3                                                  <--
mouseY = 3                                                  <--
```

Getting Player Input for Movement

At the beginning of the game we want to tell the player where the cat currently is and then take input to find out where to move the cat. You can print text to the screen to tell the player where the cat is with the print statement. Then you can use a variable to store their input. In this case we will name our variable "direction" and store the input in it.

```
print("The cat is currently at ", catX, ", ", catY)
direction = input("Which direction do you want to go? ")
```

Our code should now look like:

```
catX = 0
catY = 0

mouseX = 3
mouseY = 3

print("The cat is currently at ", catX, ", ", catY)}        <--
direction = input("Which direction do you want to go? ")    <--
```

Moving the Cat

So now we have a variable "direction" that contains the direction the player wants to move - either "l" for left, "r" for right, "d" for down, and "u" for up. For example. if the player inputs "l", then we should move the cat to the left. To move the cat to the left we need to decrease "catX" by 1. This can be accomplished with an if statement! (Don't forget to indent for the line inside of the if statement!)

```
if direction == "l":
  catX = catX - 1
```

Now, just do the same thing for all 4 directions, like so:

```
if direction == "l":
  catX = catX - 1
if direction == "r":
  catX = catX + 1
if direction == "d":
  catY = catY - 1
if direction == "u":
  catY = catY + 1
```

Our program should now look like:

```
catX = 0
catY = 0

mouseX = 3
mouseY = 3

print("The cat is currently at ", catX, ", ", catY)
direction = input("Which direction do you want to go? ")

if direction == "l":                                          <--
   catX = catX - 1                                            <--
if direction == "r":                                          <--
   catX = catX + 1                                            <--
if direction == "d":                                          <--
   catY = catY - 1                                            <--
if direction == "u":                                          <--
   catY = catY + 1                                            <--
```

Create the Gameplay Loop

We want the game to run until the cat finds the mouse - until the location of the cat and the location of the mouse are the same. To do this we are going to use a while loop which will loop until "catX" is equal to"mouseX" and "catY" is equal to "mouseY". In the context of a while loop, it will loop while the cat is not in the same position as the mouse. This can be done with the following line:

```
while not (catX == mouseX and catY == mouseY):
```

Our code should now look like this: (Don't forget to indent the contents of the loop!)

```
catX = 0
catY = 0

mouseX = 3
mouseY = 3

while not(catX == mouseX and catY == mouseY):                 <--
   print("The cat is currently at ", catX, ", ", catY)
   direction = input("Which direction do you want to go? ")

   if direction == "l":
```

```
        catX = catX - 1
    if direction == "r":
        catX = catX + 1
    if direction == "d":
        catY = catY - 1
    if direction == "u":
        catY = catY + 1
```

Let the Player Know that they Won!

Lastly, we need some way to let the player know that they won. This can be done using a print statement after the while loop, such that it will only occur once the cat has found the mouse. (Make sure this line is NOT indented)

```
print("You found the Mouse at ", mouseX, ",", mouseY)
```

And now, our finished program should look like:

```
catX = 0
catY = 0

mouseX = 3
mouseY = 3

while not(mouseX == catX and mouseY == catY):
    print("The cat is currently at ", catX, ", ", catY)
    direction = input("Which direction do you want to go? ")

    if direction == "l":
        catX = catX - 1
    if direction == "r":
        catX = catX + 1
    if direction == "d":
        catY = catY - 1
    if direction == "u":
        catY = catY + 1

print("You found the Mouse at ", mouseX, ",", mouseY)        <--
```

Go ahead, give it a play! Try changing the "mouseX" and "mouseY" values and letting a friend try to find the mouse. If anything went wrong and you can't figure it out, a working version can be found at https://onlinegdb.com/ByxP-6s6Q

Challenges

Now that you have finished the introduction, here is a list of things you can try with the code:

- Try to make it so the player automatically wins - think back to what causes the while loop finish
- Add a secret coordinate, such that if the cat goes there, it prints a message (Try to add an additional if statement inside the while loop that checks the cat's location)
- What if we wanted the mouse's starting position to be random, so that even the author does not know where the mouse could be? This can be done with a python function called "random"!

  For starters, we have to add the following code to the top of our file so that we can access the "random" function.

```
import random
```

  Then, where we declare the the location of the mouse we can change the number with a random call, like so:

```
mouseX = random.randint(0,3);
mouseY = random.randint(0,3);
```

  This will make it so that the mouse starts somewhere before 0 and 3 at random in both the x and y.

  A working example can be found at https://onlinegdb.com/SJgUpnipQ

- What if we wanted to limit the cat's movement so that it can't move too far away? We can append the if statements so that if the cat is already at the border, the cat cant move any further in that direction

```
if direction == "l" and catX > 0:
```

  This is how we would make it so the cat cant go any farther left than 0 - can you try the same for the other directions?

  A working example can be found at https://onlinegdb.com/rkXY9no67

- Now finding a mouse is fun but what if we wanted to let the player know if they are getting closer or farther? This can be done using the distance formula and a variable.

  The distance formula is: $distance = \sqrt{(b_x - a_x)^2 + (b_y - a_y)^2}$

First we need to get the initial distance from the cat to the mouse, this should be done before the while loop. In code this looks like:

```
prev_distance = math.sqrt(math.pow(catX - mouseX, 2) +
   math.pow(catY - mouseY, 2))
```

Then inside of the while loop we need to find the new distance and see if it is greater or lesser than the old distance. Then lastly, we need to set the previous distance to the new distance. The following code accomplishes this:

```
new_distance = math.sqrt(math.pow(catX - mouseX, 2) +
   math.pow(catY - mouseY, 2))

if new_distance > prev_distance:
   print("Colder")
if new_distance < prev_distance:
   print("Hotter")

prev_distance = new_distance
```

A working example can be found at https://onlinegdb.com/SyiTk6jam

- An example with all of the listed challenges can be found at https://onlinegdb.com/BkEkWasTQ
- Be sure to experiment and try to add your own changes!