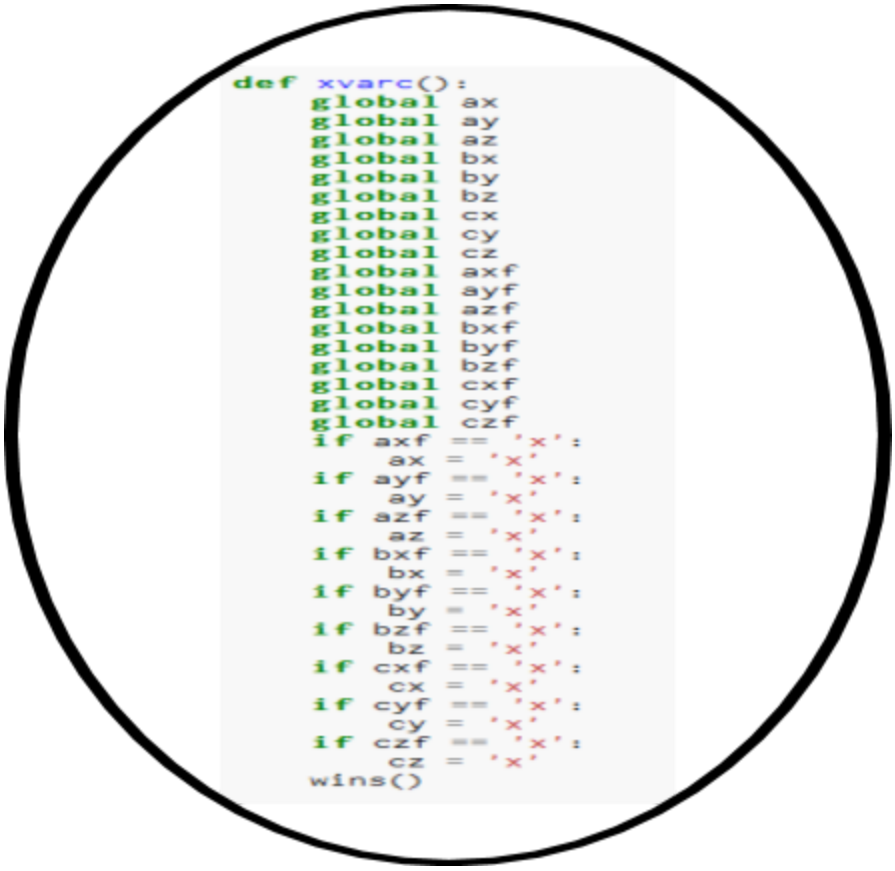
Peter N Kubas
Justin Lanfried
Computer Science
03-15-2018

Create Task Written Response

2a.) Using Python 2, I created a tic tac toe program in the canopy environment. The program runs a game a tic tac toe in which two players take turns entering coordinates to place their pieces. The values on the x-axis are a, b and c while the y-axis uses x, y and z. In relation to an actual grid, a, b and c are equal to 1, 2 and 3 respectively. However x, y and z are equal to -1, -2 and -3 respectively.

2b.) The function displayed below , xvarc(), stands for x variable correction. Essentially, the function will check to see which variables ending in f are equal to x and change the corresponding variables that do not end in f to equal x. This is a necessary solution to an obstacle I encountered whilst coding. In the xturn() and oturn() functions, I used if statements to check if the user input equaled one of the board positions and if that position was currently empty (equal to '_'). If both were true then the the selected position variable would change to be equal to 'x'. However, I discovered that in python, variables cannot be called in an if statement and then have their value changed under that statement. Therefore, I had to create a duplicate set of variables. I used the duplicate set in place of the first under the if statements in the xturn and oturn functions. Then in the xvarc() and ovarc() functions, the original set of variables is set to reflect the duplicate, therefore solving my problem.



```
def xvarc():
    global ax
    global ay
    global az
    global bx
    global by
    global bz
    global cx
    global cy
    global cz
    global axf
    global ayf
    global azf
    global bxf
    global byf
    global bzf
    global cxf
    global cyf
    global czf
    if axf == 'x':
        ax = 'x'
    if ayf == 'x':
        ay = 'x'
    if azf == 'x':
        az = 'x'
    if bxf == 'x':
        bx = 'x'
    if byf == 'x':
        by = 'x'
    if bzf == 'x':
        bz = 'x'
    if cxf == 'x':
        cx = 'x'
    if cyf == 'x':
        cy = 'x'
    if czf == 'x':
        cz = 'x'
    wins()
```

This is the function play(), which serves as my abstraction. play() contains all my functions, making it easy to call them in order. One obstacle I encountered however, is that the the function wouldn't stop even if a winner had already been decided. This means that even if x had won, it would still ask o to take their turn. To remedy this, I split x and o into two different if statements that would run if winner = 'none'. Furthermore, I encased them in a while loop. The loop runs as long as Pewter = 'seraphim'. Therefore, beneath each if statement, I implemented an else statement that set Pewter = 'demon', terminating the while loop and ending the game. The else statement would only be called if a winner had been changed to 'x' or 'o', ensuring that if x won, o wouldn't get another turn.

```
214  def play():
215      Pewter = 'seraphim'
216      while Pewter == 'seraphim':
217          if winner == 'none':
218              print "It is x's turn! enter a coordinate to play a piece."
219              xturns()
220              xvarc()
221          else:
222              Pewter = 'demon'
223          if winner == 'none':
224              print "It is o's turn! enter a coordinate to play a piece."
225              oturns()
226              ovarc()
227          else:
228              Pewter = 'demon'
```