Tic-Tac-Toe Program Steps

April 9, 2020

1 Lists,Loops and Functions on Python

[2]:
```
programming_languages = "Python","C","C++", "Java"
print(type(programming_languages))
print(programming_languages)
#here we use tuple tuples cannot be changed , lists can be changed
```

```
<class 'tuple'>
('Python', 'C', 'C++', 'Java')
```

[19]:
```
game = [ [1,0,0],
         [0,1,0],
         [0,0,1],
       ]
print(game) # it prints all the list
for rows in game: #to print above of each one
    print(rows)
```

```
[[1, 0, 0], [0, 1, 0], [0, 0, 1]]
[1, 0, 0]
[0, 1, 0]
[0, 0, 1]
```

[6]:
```
seasons = ["Spring","Summer","Autumn","Winter"]
list(enumerate(seasons))
```

[6]:
```
[(0, 'Spring'), (1, 'Summer'), (2, 'Autumn'), (3, 'Winter')]
```

[28]:
```
game = [[1,0,0],
        [0,1,0],
        [0,0,1],
       ]
print("   a  b  c ")
for count,rows in enumerate(game):
    print(count,rows)
# enumerates the list elements by starting 0 and does not affect "rows" variable.
 ↪
```

```
   a  b  c
0 [1, 0, 0]
1 [0, 1, 0]
2 [0, 0, 1]
```

[5]:
```
game = [[1,0,0],
        [0,1,0],
        [0,0,1],
        ]
print("   a  b  c ")
for count,rows in enumerate(game):
    print(count,rows)
```

```
   a  b  c
0 [1, 0, 0]
1 [0, 1, 0]
2 [0, 0, 1]
```

[21]:
```
x = [10,20,30,40,50]
print(x[0:4]) # starts 0. element and prints 4 element
print(x[2:]) # starts 2.element prints till end

game = [[1,2,3],
        [4,5,6],
        [7,8,9],
        ]
game[0][2] #here there are list of lists , it means 2th element of 0 th list.
for y in game:
    print(y)
```

```
[10, 20, 30, 40]
[30, 40, 50]
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

[26]:
```
game = [ [1,0,0],
         [0,1,0],
         [0,0,1],
        ]
def game_board():
    print("   a  b  c ")
    for count,row in enumerate(game):
        print(count,row)
game_board() #paranthesis is necessary to run the function
game[2][0] = 58
x = game_board # we are just defining not run the function so paranthesis are␣
 ↪not required here
```

```
x() # again here we call the function and make it run
```

```
   a  b  c
0 [1, 0, 0]
1 [0, 1, 0]
2 [0, 0, 1]
   a  b  c
0 [1, 0, 0]
1 [0, 1, 0]
2 [58, 0, 1]
```

[11]:
```
game = [[0,0,0],
        [0,0,0],
        [0,0,0],
       ]

def game_board(player=0, row=0,column=0, just_display=False):
    print("   0  1  2 ")
    if not just_display: #enters this conditions when user inputs
        game[row][column] = player #the move by the user
    for count , row in enumerate(game):
        print(count,row)

game_board(just_display=True)#prints the initial output which has all zero.
game_board(player = 1,row=2,column=1)
```

```
   0  1  2
0 [0, 0, 0]
1 [0, 0, 0]
2 [0, 0, 0]
   0  1  2
0 [0, 0, 0]
1 [0, 0, 0]
2 [0, 1, 0]
```

2 Mutability in Python

[8]:
```
game = "I want to play a game"
print(game)
print(id(game))  #id is equivalent the memory address in C, it is unique

def game_func():
    global game #helps us to change global variable
    game = "just play a game"
print(id(game)) # function is not called yet
print(game)
```

```
game_func()
print(id(game)) # since we change global variable in the function
print(game)
```

```
I want to play a game
1430420195728
1430420195728
I want to play a game
1430420195648
just play a game
```

3 Using return on functions

[12]:
```
def no_return(x,y):
    c = x + y
# no_return function does not return anything

res = no_return(4,5)
print(res)
```

```
None
```

[10]:
```
def no_return(x,y):
    c = x + y
    return c #function returns c value here

res = no_return(4,5)
print(res)
```

```
9
```

[24]:
```
# That program must be runned in PyCharm there is different output here last␣
 ↪output shouldnt exist
game = [ [0,0,0],
         [0,0,0],
         [0,0,0],
       ]

def game_board(game_now,player=0, row=0,column=0, just_display=False):
    print("   0  1  2 ")
    if not just_display: #enters this conditions when user inputs
        game_now[row][column] = player #the move by the user
    for count , row in enumerate(game_now):
        print(count,row)
    return game_now
```

```
game_board(game,just_display=True)
game_board(game,player = 1,row=2,column=1)
```

```
   0  1  2
0 [0, 0, 0]
1 [0, 0, 0]
2 [0, 0, 0]
   0  1  2
0 [0, 0, 0]
1 [0, 0, 0]
2 [0, 1, 0]
```

[24]:
```
[[0, 0, 0], [0, 0, 0], [0, 1, 0]]
```

[27]:
```
x = 1
def test():
    x = 2
test()
print(x) # To change global variable we must write global x


x = 1
def test():
    global x
    x = 2
test()
print(x) # global variable changed in the function


x = [1]
def test():
    x = [2]
test()
print(x) # again global variable is not changed


x = [1]
def test():
    global x
    x = [2]
test()
print(x) #[2] due to global x definition


x = [4,5,1]
def test():
    x[0] = 2
```

```
test()
print(x) # it is the proof of that elements of lists can be changed by not␣
 ↪writing global x here.
```

```
1
2
[1]
[2]
[2, 5, 1]
```

4 Error Handling in Python

[7]:
```
game = [ [0,0,0],
         [0,0,0],
         [0,0,0],
        ]

def game_board(game_now,player=0, row=0,column=0, just_display=False):
    try: # program will try using the function assignments, if user enters␣
 ↪invalid input , program shows error message by except
        print("   0  1  2 ")
        if not just_display: #enters this conditions when user inputs
            game_now[row][column] = player #the move by the user
        for count , row in enumerate(game_now):
            print(count,row)
        return game_now
    except IndexError as e: #invalid user input condition
        print("Error: make sure that you entered 0,1 or 2",e)
    except Exception as e:
        print("Something went really wrong!",e)
game_board(game,just_display=True)
game_board(game_board,player = 1,row=3,column=1)
#write game instead of game_board to observe the other error

# there is no row=3 instead of showing the error , program says make sure that␣
 ↪you entered 0,1 or 2
# when game_board is entered which does not exist , program says Something went␣
 ↪really wrong!
```

```
   0  1  2
0 [0, 0, 0]
1 [0, 0, 0]
2 [0, 0, 0]
   0  1  2
Something went really wrong! 'function' object is not subscriptable
```

[13]:
```
game = [[2,2,0],
        [2,2,2],
        [0,1,2],]

def win (current_game):
    for row in game:
        print(row)
        all_match = True
        for item in row:
            if item!= row[0]:
                all_match = False
        if all_match:
            print("winner")
win(game)

# here the code compares the row elements (horizontal)
# and decides if they are equal, prints winner if they re equal
# it is just horizontal tic tac toe program
```

```
[2, 2, 0]
[2, 2, 2]
winner
[0, 1, 2]
```

5 Checking if the horizontal elements are the same

[16]:
```
game = [[2,2,0],
        [2,2,2],
        [0,1,2],]


def win(current_game):
    for row in game:
        print(row)
        if row.count(row[0]) == len(row) and row[0] != 0:
            print("Winner!")
win(game)

#x.count(x[0]) == len(x) is a special function to compare elements of row
```

```
[2, 2, 0]
[2, 2, 2]
Winner!
[0, 1, 2]
```

6 Checking if the vertical elements are the same

[2]:
```
game = [[2,1,1],
        [2,1,2],
        [1,1,1],
        ]

for col in range(len(game)): # it is like a basic for loop starts counting 0 to 3
    check = [] # it is for assigning column elements and check if they are the␣
 ↪same.
    for row in game: # adds row elements to check one by one
        check.append(row[col]) # adds first elements of row to check list.after␣
 ↪second and third
        print(check)
    if check.count(check[0])== len(check) and check[0] !=0: # controls if check␣
 ↪has same elements to decide if game is won
        print("Winner!")
```

```
[2]
[2, 2]
[2, 2, 1]
[1]
[1, 1]
[1, 1, 1]
Winner!
[1]
[1, 2]
[1, 2, 1]
```

7 Checking if the diagonal elements are the same

[10]:
```
game = [[1,2,1],
        [2,1,2],
        [2,1,1],
        ]

diag = []

for ix in range(len(game)):
    diag.append(game[ix][ix])
    print(diag)
if diag.count(diag[0]) == len(diag) and diag[0] != 0:
       print("Winner!")
```

```
[1]
[1, 1]
```

```
[1, 1, 1]
Winner!
```

[3]:
```
#as seen below , we can use enumerate instead of zip here
game = [[1,2,1],
        [2,1,2],
        [1,1,1],
        ]

cols = reversed(range(len(game)))
rows = range(len(game))
diag = []
for col,row in zip(cols,rows):
    print(col,row)
    diag.append(game[row][col])
    print(diag)

if diag.count(diag[0]) == len(diag) and diag[0] != 0:
       print("Winner!")
```

```
2 0
[1]
1 1
[1, 1]
0 2
[1, 1, 1]
Winner!
```

[8]:
```
game = [[1,2,1],
        [2,1,2],
        [1,1,1],
        ]

diag = []
for col,row in enumerate(reversed(range(len(game)))):
    print(col,row)
    diag.append(game[row][col])
    print(diag)

if diag.count(diag[0]) == len(diag) and diag[0] != 0:
        print("Winner!")
```

```
0 2
[1]
1 1
[1, 1]
2 0
[1, 1, 1]
```

```
Winner!
```

[11]:
```
#this code make us jump to next element every time
import itertools

x = [1,2,3,4] # iterable

n= itertools.cycle(x) # iterator also iterable

y= iter(x)# iterator also iterable

next(y) #makes y = 2

for i in y:
    print(i)
for i in y:
    print(i)
```

```
2
3
4
```

[12]:
```
# Here we want to display 0 1 2
game_size = 3
print("   0  1  2")
s = "   "+"  ".join([str(i) for i in range(game_size)])
# here it can be printed by two ways as shown
print(s)
```

```
   0  1  2
   0  1  2
```

[18]:
```
# dictionaries

dictionaries = {"key1":15,"key2":32}
print(dictionaries["key1"])
dictionaries["hithere"] = 92
print(dictionaries)
```

```
15
{'key1': 15, 'key2': 32, 'hithere': 92}
```

[20]:
```
game_size = int(input("What size of tic tac toe do you want to play?:"))
game = [[0 for i in range(game_size)] for i in range(game_size)]
print(game)
```

```
What size of tic tac toe do you want to play?:3
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

```
[ ]: # Please copy this code to any compiler to see damn colours

import itertools
from colorama import Fore,Back,Style,init
init()
def win(current_game):
    def all_same(l):
        if l.count(l[0]) == len(l) and l[0] != 0:
            return True
        else:
            return False

    for row in game:
        #print(row)
        if all_same(row):
            print(f"Player {row[0]} is the Winner,horizontally!")
            return True

        diag = []
        for col, row in enumerate(reversed(range(len(game)))):
            diag.append(game[row][col])
        if all_same(diag):
            print(f"Player {diag[0]} is the winner diagonally(/)!")
            return True

    diag = []

    for ix in range(len(game)):
        diag.append(game[ix][ix])
        #print(diag)
    if all_same(diag):
        print(f"Player {diag[0]} is the Winner,diagonally  (\\)!")
        return True

    for col in range(len(game)):  # it is like a basic for loop starts counting␣
↪0 to 3
        check = []  # it is for assigning column elements and check if they are␣
↪the same.
        for row in game:  # adds row elements to check one by one
            check.append(row[col])  # adds first elements of row to check list.
↪after second and third
        if all_same(check) : # controls if check has same elements to decide if␣
↪game is won
            print(f"Player {check[0]} is the winner verticially!")
            return True
    return False
```

```
def game_board(game_map,player=0, row=0,column=0, just_display=False):
    try:# program will try using the function assignments, if user enters␣
 ↪invalid input , program shows error message by except7
        if game_map[row][column] != 0:
            print("This position is occupado! Choose another!")
            return game_map,False
        print("   "+"  ".join([str(i) for i in range(len(game_map))]))
        if not just_display: #enters this conditions when user inputs
            game_map[row][column] = player #the move by the user
        for count , row in enumerate(game_map):
            colored_row =""
            for item in row:
                if item ==0:
                    colored_row += "  "
                elif item ==1 :
                    colored_row += Fore.GREEN + ' X ' + Style.RESET_ALL
                elif item ==2 :
                    colored_row += Fore.MAGENTA + ' O ' + Style.RESET_ALL
            print(count,colored_row)


        return game_map,True

    except IndexError as e: #invalid user input condition
        print("Error: make sure that you entered 0,1 or 2",e)
        return game_map,False
    except Exception as e:
        print("Something went really wrong!",e)
        return game_map,False
play= True
players = [1,2]
while play:
    game = [[0, 0, 0],
            [0, 0, 0],
            [0, 0, 0],
            ]
    game_won = False
    game, _ = game_board(game,just_display=True)
    player_choice = itertools.cycle([1,2])
    while not game_won:
        current_player = next(player_choice)
        print(f"Current player:{current_player}")
        played = False

        while not played:
```

```
            column_choice = int(input("What column do you want to play? (0,1,2):
↪"))
            row_choice = int(input("What row do you want to play? (0,1,2):"))
            game,played =␣
↪game_board(game,current_player,row_choice,column_choice)
        if win(game):
            game_won = True
            again = input("The game is over would you like to play again? (y/n)")
            if again.lower() == "y":
                print("restarting")
            elif again.lower() == "n":
                print("Bye then")
                play = False
            else :
                print("Invalid answer see you later")
                play = False
```

```
   0  1  2
0
1
2
Current player:1
What column do you want to play? (0,1,2):0
What row do you want to play? (0,1,2):0
   0  1  2
0  X
1
2
Current player:2
What column do you want to play? (0,1,2):1
What row do you want to play? (0,1,2):1
   0  1  2
0  X
1     O
2
Current player:1
```

[ ]: