**Outline**

Access the Python Development environment and continue the tutorial to gain an additional exposure to the Python programming language. Begin to develop an familiarity with intermediate programming concepts.

**Objectives**

* Use correct terminology to describe programming concepts;
* Describe the types of data that computers can process and store (e.g., numbers, text);
* Explain the difference between constants and variables used in programming;
* Use variables, expressions, and assignment statements to store and manipulate numbers and text in a program

**Materials**

* Python3 Development Environment at: //repl.it/
* Python Tutorial at: <http://www.letslearnpython.com/learn/>

**Accessing the Tutorial**

Accessing the Tutorial

* Go to: <http://www.letslearnpython.com/learn/>
* Read up to “Lesson 12: Input”

**Level 1: Input & Output**

1. Read through “Lesson 12: Input – What Is Input?” and “Lesson 12: Input – Example” and “Lesson 12: Input – Shortcut”.
2. Type the following code into the white area of the IDE and run the program. Explain what you see in the black area of the IDE.

print("Type your name:")

name = input()

print("Hi", name, "how are you?")

Basically what ever you input , it will output as “Hi” Input” How are you”. So for example you input John, the program will output Hi John how are you.

1. Create a short program that reads numerical input from the console and does the following:
   1. Uses the input() function to read a numerical value from the console.
   2. Calculates the square root of the number
   3. Prints the result to the console output
   4. Provides appropriate prompt and message strings to go with the input and output.
   5. Provide your complete program below.
2. print("Which number do you want to square root")
3. sr=int(input())
4. import math
5. ar=(math.sqrt(sr))
6. print(ar)

**Level 2: Tic-Tac-Toe Game**

1. Write a Python program to play a game of Toc-Tac-Toe. (You may modify a program that you found on-line to meet the expectations of this module.)
   1. The program may be either player v. computer or player 1 v. player 2.
   2. The program does not need to determine a winner
   3. The program just needs to keep track of moves and spaces in the game board
2. Provide a complete listing of your program.
   1. Your listing **MUST** include line numbers .
3. print("Welcome to Tic Tac toe, you will be 'X' and the computer will be 'O'")
4. x=range(0,100)
5. import random
6. board = ['\_'] \* 10000
7. def print\_board():
8. print(board[0] + '|' + board[1] + '|' + board[2])
9. print(board[3] + '|' + board[4] + '|' + board[5])
10. print(board[6] + '|' + board[7] + '|' + board[8])
11. print\_board()
12. while True:
13. x = input('Pick a number from 0-8 (Do not pick the same number twice): ')
14. x = int(x)
16. if x<0 or x>8:
17. print('Pick A different spot')
18. elif board[x] != 'O' and board[x] != 'X':
19. board[x] = 'X'
20. print\_board()
22. print("The Computer will take it's turn")
23. O = random.randint(0,8)
24. board[O] = "O"
25. if board[O] != 'O' and board[O] != 'X':
26. board[O] = 'O'
28. print\_board()
30. Explain how your program keeps track of the game board.   
    (Provide specific code references by line number.)
    1. What python types and data structures are used?

The code uses lists, strings(24), integers(26) and etc.

* 1. How are moves by player X and player O recorded?

Where ever the player wants to move he/she inputs the number related to which spot they want to do their move in.(15,16)

* 1. How are free spaces recorded?

If the space has nothing on it, it is a free space.

1. Explain how moves and commands are input from the console.  
   (Provide specific code references by line number.)
   1. How does the player tell the program about the move location (row, column)?

The player knows about the move location by the numbers on the spaces, the numbers represent that space so if you want to go on space 2 you input 2 and your letter is written on space 2.(15)

* 1. How does the program verify that the move location is valid?
  2. How does the program verify that the space is free?
  3. What does the program do if there is something wrong with the move?

It basically checks if the space has X or O on it and if your input is between the range. If the space does have a X or O on it or your input is not in between the integer range then it tells you to go again.(18-22)

1. Explain how the program keeps track of gameplay.  
   (Provide specific code references by line number.)
   1. How does the program switch between player X and player O moves?

Basically first the player inputs his move, then the program outputs X wherever the player wants it (20-22) then the computer picks a random integer between 0-8 and outputs an O onto that space.(26-29)

* 1. How does the program keep asking for moves?

The program is in a while loop so until a condition is met the loop will not stop and the program will keep asking for moves.(14)

* 1. How does the program decide when to stop asking for moves?

The program decides when to stop asking once a winner is decided, once a winner is decided the loop is then broken.

**Level 3: Basic Enhancements**

1. Explain, in plain words, a strategy for determining if player “x” or player “O” has won the game after a move is made.

You can check each possible combination for winning, you check if there is 3 X’s or 3 O’s in that combination, if the result is true than break the loop and if O wins print the computer has won, and if X wins print the Player has won.

1. Provide a function called “checkWinForX” that returns the Boolean value of “True” if player “x” won the game.
2. Modify your program to check and print a message, and stop the game of player “x” or player “O” wins the game.
3. def checkWin(player) :
4. if board[0] == player and board[1] == player and board[2] == player:
5. return True
6. if board[3] == player and board[4] == player and board[5] == player:
7. return True
8. if board[6] == player and board[7] == player and board[8] == player:
9. return True
10. if board[0] == player and board[3] == player and board[6] == player:
11. return True
12. if board[1] == player and board[4] == player and board[7] == player:
13. return True
14. if board[2] == player and board[5] == player and board[8] == player:
15. return True
16. if board[0] == player and board[4] == player and board[8] == player:
17. return True
18. if board[2] == player and board[4] == player and board[6] == player:
19. return True
20. if checkWin('X') == True:
21. print('You Beat the Computer')
22. break;
23. Demonstrate your enhanced game to Mr. Nestor for credit for this level.

**Level 4: AI Enhancements**

1. Explain, in plain words, a strategy for suggesting the best move for player “x” or player “O” to make when it is their turn to move.

Basically you check if there is a possible win combination, and suggest the player to place his move their to win. For example, you check if board 1 and 2 are x if they are you tell the player to place it on x to win.

1. Create a function to implement your strategy for suggesting the best move.

def ai():

if board[0] and board[1] =="X" and board[2] != 'O' and board[2] != 'X':

O=2

return O

elif board[1] and board[2] =="X" and board[0] != 'O' and board[0] != 'X' :

O=0

return O

elif board[3] and board[4] =="X" and board[5] != 'O' and board[5] != 'X' :

O=5

return O

elif board[5] and board[4]=="X" and board[3] != 'O' and board[3] != 'X':

O=3

return O

elif board[6] and board[7]=="X" and board[8] != 'O' and board[8] != 'X' :

O=8

return O

elif board[8] and board[7]=="X" and board[6] != 'O' and board[6] != 'X' :

O=6

return O

elif board[0] and board[2]=="X" and board[1] != 'O' and board[1] != 'X' :

O=1

return O

elif board[3] and board[5]=="X" and board[4] != 'O' and board[4] != 'X' :

O=4

return O

elif board[6] and board[8]=="X" and board[7] != 'O' and board[7] != 'X' :

O=7

return O

#vertical combinations

elif board[0] and board[3] =="X" and board[6] !='O' and board[6] != 'X' :

O=6

return O

elif board[3] and board[6]=="X" and board[0] != 'O' and board[0] != 'X' :

O=0

return O

elif board[1] and board[4]=="X" and board[7] != 'O' and board[7] != 'X' :

O=7

return O

elif board[7] and board[4]=="X" and board[1] != 'O' and board[1] != 'X' :

O=1

return O

elif board[2] and board[5]=="X" and board[8] != 'O' and board[8] != 'X' :

O=8

return O

elif board[8] and board[5]=="X" and board[2] != 'O' and board[2] != 'X' :

O=2

return O

elif board[1] and board[7]=="X" and board[4] != 'O' and board[4] != 'X' :

O=4

return O

elif board[0] and board[6]=="X" and board[3] != 'O' and board[3] != 'X' :

O=3

return O

elif board[2] and board[8]=="X" and board[5] != 'O' and board[5] != 'X' :

O=5

return O

#Diagonal combinations

elif board[0] and board[4]=="X" and board[8] != 'O' and board[8] != 'X' :

O=8

return O

elif board[4] and board[8]=="X" and board[0] != 'O' and board[0] != 'X' :

O=0

return O

elif board[2] and board[4]=="X" and board[6] != 'O' and board[6] != 'X' :

O=6

return O

elif board[6] and board[4]=="X" and board[2] != 'O' and board[2] != 'X' :

O=2

return O

elif board[6] and board[2]=="X" and board[4] != 'O' and board[4] != 'X' :

O=4

return O

elif board[8] and board[0]=="X" and board[4] != 'O' and board[4] != 'X' :

O=4

return O

else:

O = random.randint(0,8)

return O

This is for O only.

1. Modify your program to print a suggested move when it is each player’s turn to move.

I if board[1] and board[2] =="X":

Print("You should place on board 3 to win")

And so on for the possible combinations.

1. Demonstrate your AI enhanced game to Mr. Nestor for credit for this level.

Final Code:

def c1():

if board[O] == 'O' and board[O] == 'X':

pass

return True

else:

board[O]="O"

return False

def spot():

if x<0 or x>8:

print('Pick A different spot')

return True

else:

return False

print("Welcome to Tic Tac toe, you will be 'X' and the computer will be 'O'")

x=range(0,100)

import random

board = ['\_'] \* 10000

def print\_board():

print(board[0] + '|' + board[1] + '|' + board[2])

print(board[3] + '|' + board[4] + '|' + board[5])

print(board[6] + '|' + board[7] + '|' + board[8])

print\_board()

def checkWin(player) :

if board[0] == player and board[1] == player and board[2] == player:

return True

if board[3] == player and board[4] == player and board[5] == player:

return True

if board[6] == player and board[7] == player and board[8] == player:

return True

if board[0] == player and board[3] == player and board[6] == player:

return True

if board[1] == player and board[4] == player and board[7] == player:

return True

if board[2] == player and board[5] == player and board[8] == player:

return True

if board[0] == player and board[4] == player and board[8] == player:

return True

if board[2] == player and board[4] == player and board[6] == player:

return True

def ai():

if board[0] and board[1] =="X" and board[2] != 'O' and board[2] != 'X':

O=2

return O

elif board[1] and board[2] =="X" and board[0] != 'O' and board[0] != 'X' :

O=0

return O

elif board[3] and board[4] =="X" and board[5] != 'O' and board[5] != 'X' :

O=5

return O

elif board[5] and board[4]=="X" and board[3] != 'O' and board[3] != 'X':

O=3

return O

elif board[6] and board[7]=="X" and board[8] != 'O' and board[8] != 'X' :

O=8

return O

elif board[8] and board[7]=="X" and board[6] != 'O' and board[6] != 'X' :

O=6

return O

elif board[0] and board[2]=="X" and board[1] != 'O' and board[1] != 'X' :

O=1

return O

elif board[3] and board[5]=="X" and board[4] != 'O' and board[4] != 'X' :

O=4

return O

elif board[6] and board[8]=="X" and board[7] != 'O' and board[7] != 'X' :

O=7

return O

#vertical combinations

elif board[0] and board[3] =="X" and board[6] !='O' and board[6] != 'X' :

O=6

return O

elif board[3] and board[6]=="X" and board[0] != 'O' and board[0] != 'X' :

O=0

return O

elif board[1] and board[4]=="X" and board[7] != 'O' and board[7] != 'X' :

O=7

return O

elif board[7] and board[4]=="X" and board[1] != 'O' and board[1] != 'X' :

O=1

return O

elif board[2] and board[5]=="X" and board[8] != 'O' and board[8] != 'X' :

O=8

return O

elif board[8] and board[5]=="X" and board[2] != 'O' and board[2] != 'X' :

O=2

return O

elif board[1] and board[7]=="X" and board[4] != 'O' and board[4] != 'X' :

O=4

return O

elif board[0] and board[6]=="X" and board[3] != 'O' and board[3] != 'X' :

O=3

return O

elif board[2] and board[8]=="X" and board[5] != 'O' and board[5] != 'X' :

O=5

return O

#Diagonal combinations

elif board[0] and board[4]=="X" and board[8] != 'O' and board[8] != 'X' :

O=8

return O

elif board[4] and board[8]=="X" and board[0] != 'O' and board[0] != 'X' :

O=0

return O

elif board[2] and board[4]=="X" and board[6] != 'O' and board[6] != 'X' :

O=6

return O

elif board[6] and board[4]=="X" and board[2] != 'O' and board[2] != 'X' :

O=2

return O

elif board[6] and board[2]=="X" and board[4] != 'O' and board[4] != 'X' :

O=4

return O

elif board[8] and board[0]=="X" and board[4] != 'O' and board[4] != 'X' :

O=4

return O

else:

O = random.randint(0,8)

return O

while True:

x = input('Pick a number from 0-8 (Do not pick the same number twice): ')

x = int(x)

if spot()== True:

continue;

if board[x] != 'O' and board[x] != 'X':

board[x] = 'X'

print\_board()

if checkWin('X') == True:

print('You Beat the Computer')

break;

print("The Computer will take it's turn")

while True :

O=ai()

z=c1()

if z==False:

break;

print\_board()

if checkWin('O') == True:

print('The Computer beat YOU')

break;