**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?")

It asks for my anems, and when a name is entered, it immediately follows with “Hi (name that I entered) how are you?”.

3.       Create a short program that reads numerical input from the console and does the following:

a.    Uses the input() function to read a numerical value from the console.

b.   Calculates the square root of the number

c.    Prints the result to the console output

d.   Provides appropriate prompt and message strings to go with the input and output.

e.   Provide your complete program below.

import math

def calculate():

print("enter your calculation")

print("write a number which can be square rooted to a whole number")

print("the answer is", input(), "you are welcome")

calculate()

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

1.       Write a Python program to play a game of Tic-Tac-Toe. (You may modify a program that you found online to meet the expectations of this module.)

a.    The program may be either player v. computer or player 1 v. player 2.

b.   The program does not need to determine a winner

c.    The program just needs to keep track of moves and spaces in the game board

2.       Provide a complete listing of your program.

a.    Your listing **MUST** include line numbers .

1. def ticTacToe():

2.   board = [1, 2, 3, 4, 5, 6, 7, 8, 9]

3.   end = False

4.   winCommbinations = ((0, 1, 2), (3, 4, 5), (6, 7, 8), (0, 3, 6), (1, 4, 7), (2, 5, 8), (0, 4, 8), (2, 4, 6))

5.

6.   def draw():

7.       print(board[0], board[1], board[2])

8.       print(board[3], board[4], board[5])

9.       print(board[6], board[7], board[8])

10.       print()

11.

12.   def p1():

13.       n = chooseNumber()

14.       if board[n] == "X" or board[n] == "O":

15.           print("\nYou can't go there. Try again")

16.           p1()

17.       else:

18.           board[n] = "X"

19.

20.   def p2():

21.       n = chooseNumber()

22.       if board[n] == "X" or board[n] == "O":

23.           print("\nYou can't go there. Try again")

24.           p2()

25.       else:

26.           board[n] = "O"

27.

28.   def chooseNumber():

29.       while True:

30.           while True:

31.               a = input()

32.               try:

33.                   a = int(a)

34.                   a -= 1

35.                   if a in range(0, 9):

36.                       return a

37.                   else:

38.                       print("\nThat's not on the board. Try again")

39.                       continue

40.               except ValueError:

41.                  print("\nThat's not a number. Try again")

42.                  continue

43.

44.   def checkBoard():

45.       count = 0

46.       for a in winCommbinations:

47.           if board[a[0]] == board[a[1]] == board[a[2]] == "X":

48.               print("Player 1 Wins!\n")

49.               return True

50.

51.           if board[a[0]] == board[a[1]] == board[a[2]] == "O":

52.               print("Player 2 Wins!\n")

53.               return True

54.       for a in range(9):

55.           if board[a] == "X" or board[a] == "O":

56.               count += 1

57.           if count == 9:

58.               print("DRAW\n")

59.               return True

60.

61.   while not end:

62.       draw()

63.       end = checkBoard()

64.       if end == True:

65.           break

66.       print("Player 1 choose where to place a X")

67.       p1()

68.       print()

69.       draw()

70.       end = checkBoard()

71.       if end == True:

72.           break

73.       print("Player 2 choose where to place a O")

74.       p2()

75.       print()

76.

77.   if input("Play again (y/n)\n") == "y":

78.       print()

79.       ticTacToe()

80. ticTacToe()

The code above is what I had found in the internet, due to level 1 and 2 being due, this is what will be temporarily shown in my repository. I will modify it to include the code we have went over when level 3 and 4 are due. But for now this is what I had done my work based off of.

3.       Explain how your program keeps track of the game board.

(Provide specific code references by line number.)

a.    What python types and data structures are used?

It is a logical data structure as it follows booleans via either an action being true or false. An example would be the lines 28 - 42, where certain commands only activate if conditions are true.

b.   How are moves by player X and player O recorded?

A player is given a prompt to make their move, in which they type an available number between 1-9, if the number is available, the gameboard will update with that number being occupied in theri symbol.

An example are the lines 12 and 13 or 20 and 21.

c.    How are free spaces recorded?

If a number is already in use, the player will be prompted with a message. This is shown in the lines 22, and 23 or 14 and 15.

4.       Explain how moves and commands are input from the console.

(Provide specific code references by line number.)

a.    How does the player tell the program about the move location (row, column)?

The player, during their turn, writes a number. Shown lines 13 or 21.

b.   How does the program verify that the move location is valid?

It checks if the move you requested is of the available list. If not it will give a prompt that the move is invalid. As shown in lines 28 - 42.

c.    How does the program verify that the space is free?

It will check which spaces are already used, and which spaces are left. Shown in lines 12 - 26.

d.   What does the program do if there is something wrong with the move?

It will prompt the player of the move is invalid. Shown in lines 15, 23, 38, and 41.

5.       Explain how the program keeps track of gameplay.

(Provide specific code references by line number.)

a.    How does the program switch between player X and player O moves?

The program has code written who goes first, what input they put in, and when their turn ends, into which it will break, and move onto the net players turn, and loop until the game is over. As shown in lines 61 - 75.

b.   How does the program keep asking for moves?

The loop statement written makes it so while applicable, a player get to move until a loop break condition is met, which is when one players wins or there are no more available spaces. As shown in lines 61 - 75.

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

The loop break statement must be fulfilled, which is when the game is over. As shown in lines 61 - 75.

**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.

2.       Provide a function called “checkWinForX” that returns the Boolean value of “True” if player “x” won the game.

3.       Modify your program to check and print a message, and stop the game of player “x” or player “O” wins the game.

4.       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.

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

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

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

mySTR=input(“Enter a move between 1 to 9:”)

myNum= int(input(“Enter a move between 1 to 9:”))