**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?")

When you type it in and run the program it first asks for you to type your name as the input. Then once you put your name is will print your name into the sentence to replace “name”.

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. import math
3. number = int(input("Enter a number to find square root :"))
4. if number < 0 :
5. print("Please enter a valid number.")
6. else :
7. print("Square root of {} is {} ". format(number,math. sqrt(number)))

**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 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. def tic\_tac\_toe():
4. board = [1, 2, 3, 4, 5, 6, 7, 8, 9]
5. end = False
6. win\_commbinations = ((0, 1, 2), (3, 4, 5), (6, 7, 8), (0, 3, 6), (1, 4, 7)

, (2, 5, 8), (0, 4, 8), (2, 4, 6))

1. def draw():
2. print(board[0], "/", board[1], "/", board[2])
3. print("---------")
4. print(board[3], "/", board[4], "/", board[5])
5. print("---------")
6. print(board[6], "/", board[7], "/", board[8])
7. print()
8. def p1():
9. n = choose\_number()
10. if board[n] == "X" or board[n] == "O":
11. print("You can't go there. Try again")
12. p1()
13. else:
14. board[n] = "X"
15. def p2():
16. n = choose\_number()
17. if board[n] == "X" or board[n] == "O":
18. print("You can't go there. Try again")
19. p2()
20. else:
21. board[n] = "O"
22. def choose\_number():
23. while True:
24. while True:
25. a = input()
26. try:
27. a = int(a)
28. a -= 1
29. if a in range(0, 9):
30. return a
31. else:
32. print("That's not on the board. Try again")
33. continue
34. except ValueError:
35. print("That's not a number. Try again")
36. continue
37. def check\_board():
38. count = 0
39. for a in win\_commbinations:
40. if board[a[0]] == board[a[1]] == board[a[2]] == "X":
41. print("Player 1 Wins!")
42. print("Congratulations!")
43. return True
44. if board[a[0]] == board[a[1]] == board[a[2]] == "O":
45. print("Player 2 Wins!")
46. print("Congratulations!")
47. return True
48. for a in range(9):
49. if board[a] == "X" or board[a] == "O":
50. count += 1
51. if count == 9:
52. print("The game ends in a Tie")
53. return True
54. while not end:
55. draw()
56. end = check\_board()
57. if end == True:
58. break
59. print("Player X choose where to place the cross")
60. p1()
61. print()
62. draw()
63. end = check\_board()
64. if end == True:
65. break
66. print("Player O choose where to place the circle")
67. p2()
68. print()
69. if input("Play again (y/n)") == "y":
70. print()
71. tic\_tac\_toe()
72. tic\_tac\_toe()
73. 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?

Def, if, while true, else, input, try, print, continue, in range, except, break, and return true.

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

Whenever a player picks their spot for their turn they will pick a number that hasn’t been used yet on the board. The board is made up if numbers from 1-9. Whenever it is your turn you type one of the numbers that are free and that is where the X or O will go.

* 1. How are free spaces recorded?

Whenever you take a turn your letter will take the place of a number. When there is a number that hasn’t been used yet it just stays at whatever number is assigned in its place.

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)?

At the start of the game there will be an empty board that has the number of each place recorded already. Each place is represented by a number so when they want to move to a certain they just type in the number (ex. 1,2,3,4,5,6,7,8,9). From line 6-12 be board is shown. There you can see all the possible places to put your X or O.

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

The only valid moves are the numbers that range from 1-9 and haven’t been taken. Anything other than 1-9 will be invalid and the player will be asked to try again.

* 1. How does the program verify that the space is free?

Each place has a number assigned to it from the start of the game and every time X or O makes a move the number will be replaced with the symbol. So if the place hasn’t been used yet then it will still have its number.

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

It will say that it’s an invalid move and the player will be asked to try again. From line 14-28 there is coding that tracks the players moves and if the spot is already taken then the program will say that the spot is already takes.

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?

From 14-28 there is coding that keeps track of the movement and places the players icon on the said place.

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

From 14-43 there is coding for the movement and what places they can and can’t play on and that will loop until someone wins and gets a winning pattern which is from 45-62.

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

It stops asking when one of the players get a winning move.

**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.

Make a code that lists all the possible winning patterns so when one of the players gets the pattern they will win.

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. 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.