**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”.

Done

1. 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?")

Type your name:

Manvir

Hi Manvir how are you?

 Horrible I have a headache

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.

12. def p1():

13. n = choose\_number()

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

15. print("\nYou can't go there thats not on the board. Try again")

16. p1()

17. else:

18. board[n] = "X"

**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.

done

* 1. The program does not need to determine a winner

done

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

done

1. Provide a complete listing of your program.
   1. Your listing **MUST** include line numbers .

1.def game():

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

3. end = False

4. 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))

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 = choose\_number()

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 = choose\_number()

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 choose\_number():

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 check\_board():

45. count = 0

46. for a in win\_commbinations:

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

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

49. print("Congratulations!\n")

50. return True

51.

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

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

54. print("Congratulations!\n")

55. return True

56. for a in range(9):

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

58. count += 1

59. if count == 9:

60. print("The game ends in a Tie\n")

61. return True

62.

63. while not end:

64. draw()

65. end = check\_board()

66. if end == True:

67. break

68. print("Player 1 choose where to place a cross")

69. p1()

70. print()

71. draw()

72. end = check\_board()

73. if end == True:

74. break

75. print("Player 2 choose where to place a nought")

76. p2()

77. print()

78.

79. if input("Do you want to Play again (yes/no)\n") == "yes":

80. print()

81. game()

82.

83.game()

1. 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?

List and set

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

The players moves are recorded by placing X or O instead of the number the player typed in.

12. def p1():

13. n = choose\_number()

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

15. print("\nYou can't go there thats not on the board. Try again")

16. p1()

17. else:

18. board[n] = "X"

* 1. How are free spaces recorded?

The free places keep their number until there chosen by player (so if there is a number the space is free).

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 is given the choose a number command,

12.  def p1():

13.       n = choose\_number()

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

15.           print("\nYou can't go there thats not on the board. Try again")

16.           p1()

17.       else:

18.           board[n] = "X"

, and the choice of the 9 number on the board,

(1, 2, 3)

(4, 5, 6)

(7, 8, 9)

()

Player 1 choose where to place a cross

,and so which ever number the player choses gets a X or a O depend on which player they are.

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

The game checks if the answer is in between the range of (0,9) if not it replays with “You can't go there thats not on the board. Try again”.

20. def p2():

21. n = choose\_number()

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

23. print("\nYou can't go there thats not on the board. Try again")

24. p2()

25. else:

26. board[n] = "O"

27.

28. def choose\_number():

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

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

The program checks for a X or a O and and if it sees it it replays with “You can't go there that’s not available on the board. Try again”

20. def p2():

21. n = choose\_number()

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

23. print("\nYou can't go there thats not available on the board. Try again")

24. p2()

25. else:

26. board[n] = "O"

27.

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

If the move is wrong the game replays with

15. print("\nYou can't go there thats not on the board. Try again")

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?

The game checks the players if right it records it.

12. def p1():

13. n = choose\_number()

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

15. print("\nYou can't go there thats not on the board. Try again")

16. p1()

17. else:

18. board[n] = "X"

19.

20. def p2():

21. n = choose\_number()

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

23. print("\nYou can't go there thats not on the board. Try again")

24. p2()

25. else:

26. board[n] = "O"

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

The asks the player for which number they want their symbol to be on.

12. def p1():

13. n = choose\_number()

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

The game stops asking for the move when it sees one of these combinations played by one player

4. win\_combinations = ((0, 1, 2), (3, 4, 5), (6, 7, 8), (0, 3, 6), (1, 4, 7), (2, 5, 8), (0, 4, 8), (2, 4, 6))

or if the game is a tie

56. for a in range(9):

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

58. count += 1

59. if count == 9:

60. print("The game ends in a Tie\n")

61. return True

62.

63. while not end:

64. draw()

65. end = check\_board()

66. if end == True:

67. break

68. print("Player 1 choose where to place a cross")

69. p1()

70. print()

71. draw()

72. end = check\_board()

73. if end == True:

74. break

75. print("Player 2 choose where to place a nought")

76. p2()

77. print()

<https://codereview.stackexchange.com/questions/108738/python-tic-tac-toe-game>

**Level 3: Enhancements**

t.b.d.