[**https://docs.google.com/document/u/1/?authuser=1&usp=docs\_web**](https://docs.google.com/document/u/1/?authuser=1&usp=docs_web)**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 says “Type your name:”. After I type my name it says “Hi Daksha 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.

import math

def Root():

print(“ What is number that you want to find the Square Root of?”)

num = int(input())

print(“The Square root of,num,”is”,math.sqrt(num))

Root()

**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. import random
4. #board design
5. board = ["-", "-", "-",
6. "-", "-", "-",
7. "-", "-", "-"]
8. #used in code for convience
9. game\_still\_going = True
10. winner = None
11. current\_player = "X"
12. #How the game works
13. def play\_person():
14. display\_board()
15. while game\_still\_going:
    1. handle\_turn(current\_player)
    2. check\_if\_game\_over()
    3. flip\_player()
17. if winner == "X" or winner == "O":
    1. print(winner + " has won!")
18. elif winner == None:
    1. print("It is a Tie.")
19. def play\_ai():
20. display\_board()
21. while game\_still\_going:
    1. handle\_turn(current\_player)
    2. check\_if\_game\_over()
    3. flip\_player()
23. if winner == "X" or winner == "O":
    1. print(winner + " has won!")
24. elif winner == None:
    1. print("It is a Tie.")
25. #Shows the board
26. def display\_board():
27. print("\n")
28. print(board[0] + " | " + board[1] + " | " + board[2] + " 1 | 2 | 3")
29. print("━━━━━━━━━━"," ━━━━━━━━━━")
30. print(board[3] + " | " + board[4] + " | " + board[5] + " 4 | 5 | 6")
31. print("━━━━━━━━━━"," ━━━━━━━━━━")
32. print(board[6] + " | " + board[7] + " | " + board[8] + " 7 | 8 | 9")
33. print("\n")
34. #checks if game is over
35. def check\_if\_game\_over():
36. check\_for\_winner()
37. check\_for\_tie()
38. #checks for a winner or a tie
39. def check\_for\_winner():
40. global winner
42. row\_winner = check\_rows()
43. column\_winner = check\_columns()
44. diagonal\_winner = check\_diagonals()
46. if row\_winner:
    1. winner = row\_winner
47. elif column\_winner:
    1. winner = column\_winner
48. elif diagonal\_winner:
    1. winner = diagonal\_winner
49. else:
    1. winner = None
50. def check\_for\_tie():
52. global game\_still\_going
53. if "-" not in board:
    1. game\_still\_going = False
    2. return True
54. else:
    1. return False
55. #checks rows, columns, and diagonals for a win or tie
56. def check\_rows():
57. global game\_still\_going
58. row\_1 = board[0] == board[1] == board[2] != "-"
59. row\_2 = board[3] == board[4] == board[5] != "-"
60. row\_3 = board[6] == board[7] == board[8] != "-"
61. if row\_1 or row\_2 or row\_3:
    1. game\_still\_going = False
62. if row\_1:
    1. return board[0]
63. elif row\_2:
    1. return board[3]
64. elif row\_3:
    1. return board[6]
65. else:
    1. return None
66. def check\_columns():
67. global game\_still\_going
68. column\_1 = board[0] == board[3] == board[6] != "-"
69. column\_2 = board[1] == board[4] == board[7] != "-"
70. column\_3 = board[2] == board[5] == board[8] != "-"
71. if column\_1 or column\_2 or column\_3:
    1. game\_still\_going = False
72. if column\_1:
    1. return board[0]
73. elif column\_2:
    1. return board[1]
74. elif column\_3:
    1. return board[2]
75. else:
    1. return None
76. def check\_diagonals():
77. global game\_still\_going
78. diagonal\_1 = board[0] == board[4] == board[8] != "-"
79. diagonal\_2 = board[2] == board[4] == board[6] != "-"
80. if diagonal\_1 or diagonal\_2:
    1. game\_still\_going = False
81. if diagonal\_1:
    1. return board[0]
82. elif diagonal\_2:
    1. return board[2]
84. else:
    1. return None
85. #flips and handles player turns or ai turns
86. def flip\_player():
87. global current\_player
88. if current\_player == "X":
    1. current\_player = "O"
89. elif current\_player == "O":
    1. current\_player = "X"
90. def handle\_turn(player):
91. print(player + ", it's your turn.")
92. position = input("Choose a position from 1-9: ")
93. valid = False
94. while not valid:
    1. while position not in ["1", "2", "3", "4", "5", "6", "7", "8", "9"]:
95. position = input("Choose a position from 1-9: ")
    1. position = int(position) - 1
    2. if board[position] == "-":
96. valid = True
    1. else:
97. print("You can't go there. Go again.")
98. board[position] = player
99. display\_board()
100. #asks the player to select a gamemode
101. def selectgamemode():
102. print("\n")
103. input("Would you like to play against the computer? (Yes/No):")
105. if input == "Yes" or 'yes' or 'Y' or 'y':
     1. return play\_ai()
106. if input == "No" or 'no' or 'n' or 'N':
     1. return play\_person()
107. selectgamemode()
108. **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?**
     2. **How are moves by player X and player O recorded?**
     3. **How are free spaces recorded?**

I keep track of the board by using a list (line 3) and assigning values to it (line 40-58)

1. I use integers, strings, and booleans

b & c . In line 115 i use the handle\_turn(player) function to allow the player to choose spots on the board and line 125 checks for empty spaces being on the board or not.

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)?**
   2. **How does the program verify that the move location is valid?**
   3. **How does the program verify that the space is free?**
   4. **What does the program do if there is something wrong with the move?**
2. the player enters the number within the given index of 1 and 9 to mark a cross (line 40, def display\_board, and the handle\_turn function)
3. In line 124 if the spot selected is empty it is classified as a valid option
4. In line 124 it checks if the spot hasn’t been selected by checking if it is still “-” instead of having an X or O
5. In Line 126 the program simply states “You can’t go there. Try again.”
6. **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?**
   2. **How does the program keep asking for moves?**
   3. **How does the program decide when to stop asking for moves?**
7. I use the flip\_player function in line 110 to flip between players every turn
8. I use the handle\_turn function in line 115 to ask the user to enter a value until the game ends
9. I use the check\_if\_game\_over function in line 50 to see if the game has ended in a tie or win, if not, the program will continue to ask for moves.

**Level 3: Enhancements**

t.b.d.