**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”.

Input is information we pass to a function so that we can do something with it.

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?")

I typed my name and then the code followed was, “Hi Pritpal 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

sr=int(input())

ar=(math.sqrt(sr))

print(ar)

**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. import random
4. def drawBoard(board):

7. print(' | |')
8. print(' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])
9. print(' | |')
10. print('-----------')
11. print(' | |')
12. print(' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])
13. print(' | |')
14. print('-----------')
15. print(' | |')
16. print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
17. print(' | |')
18. def inputPlayerLetter():
20. letter = ''
21. while not (letter == 'X' or letter == 'O'):
22. print('Do you want to be X or O?')
23. letter = input().upper()
24. if letter == 'X':
25. return ['X', 'O']
26. else:
27. return ['O', 'X']
28. def whoGoesFirst():
29. if random.randint(0, 1) == 0:
30. return 'computer'
31. else:
32. return 'player'
33. def playAgain():
34. print('Do you want to play again? (yes or no)')
35. return input().lower().startswith('y')
36. def makeMove(board, letter, move):
37. board[move] = letter
38. def isWinner(bo, le):
39. return ((bo[7] == le and bo[8] == le and bo[9] == le) or
40. (bo[4] == le and bo[5] == le and bo[6] == le) or
41. (bo[1] == le and bo[2] == le and bo[3] == le) or
42. (bo[7] == le and bo[4] == le and bo[1] == le) or
43. (bo[8] == le and bo[5] == le and bo[2] == le) or
44. (bo[9] == le and bo[6] == le and bo[3] == le) or
45. (bo[7] == le and bo[5] == le and bo[3] == le) or
46. (bo[9] == le and bo[5] == le and bo[1] == le))
47. def getBoardCopy(board):
48. dupeBoard = []
49. for i in board:
50. dupeBoard.append(i)
51. return dupeBoard
52. def isSpaceFree(board, move):
53. return board[move] == ' '
54. def getPlayerMove(board):
55. move = ' '
56. while move not in '1 2 3 4 5 6 7 8 9'.split() or not isSpaceFree(board, int(move)):
57. print('What is your next move? (1-9)')
58. move = input()
59. return int(move)
60. def chooseRandomMoveFromList(board, movesList):
61. possibleMoves = []
62. for i in movesList:
63. if isSpaceFree(board, i):
64. possibleMoves.append(i)
65. if len(possibleMoves) != 0:
66. return random.choice(possibleMoves)
67. else:
68. return None
69. def getComputerMove(board, computerLetter):
70. if computerLetter == 'X':
71. playerLetter = 'O'
72. else:
73. playerLetter = 'X'
74. for i in range(1, 10):
75. copy = getBoardCopy(board)
76. if isSpaceFree(copy, i):
77. makeMove(copy, computerLetter, i)
78. if isWinner(copy, computerLetter):
79. return i
80. for i in range(1, 10):
81. copy = getBoardCopy(board)
82. if isSpaceFree(copy, i):
83. makeMove(copy, playerLetter, i)
84. if isWinner(copy, playerLetter):
85. return i
86. move = chooseRandomMoveFromList(board, [1, 3, 7, 9])
87. if move != None:
88. return move
89. if isSpaceFree(board, 5):
90. return 5
91. return chooseRandomMoveFromList(board, [2, 4, 6, 8])
92. def isBoardFull(board):
93. for i in range(1, 10):
94. if isSpaceFree(board, i):
95. return False
96. return True
97. print('Welcome to Tic Tac Toe!')
98. while True:
100. theBoard = [' '] \* 10
101. playerLetter, computerLetter = inputPlayerLetter()
102. turn = whoGoesFirst()
103. print('The ' + turn + ' will go first.')
104. gameIsPlaying = True
105. while gameIsPlaying:
106. if turn == 'player':
107. drawBoard(theBoard)
108. move = getPlayerMove(theBoard)
109. makeMove(theBoard, playerLetter, move)
110. if isWinner(theBoard, playerLetter):
111. drawBoard(theBoard)
112. print('Hooray! You have won the game!')
113. gameIsPlaying = False
114. else:
115. if isBoardFull(theBoard):
116. drawBoard(theBoard)
117. print('The game is a tie!')
118. break
119. else:
120. turn = 'computer'
121. else:
122. move = getComputerMove(theBoard, computerLetter)
123. makeMove(theBoard, computerLetter, move)
124. if isWinner(theBoard, computerLetter):
125. drawBoard(theBoard)
126. print('The computer has beaten you! You lose.')
127. gameIsPlaying = False
128. else:
129. if isBoardFull(theBoard):
130. drawBoard(theBoard)
131. print('The game is a tie!')
132. break
133. else:
134. turn = 'player'
135. if not playAgain():
136. break
137. 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?
138. def getPlayerMove(board):
139. move = ' '
140. while move not in '1 2 3 4 5 6 7 8 9'.split() or not isSpaceFree(board, int(move)):
141. print('What is your next move? (1-9)')
142. move = input()
143. return int(move)

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

83. board[move]=computer

84. board[move]=player

* 1. How are free spaces recorded?

1. def chooseRandomMoveFromList(board, movesList):
2. possibleMoves = []
3. for i in movesList:
4. if isSpaceFree(board, i):
5. possibleMoves.append(i)
6. if len(possibleMoves) != 0:
7. return random.choice(possibleMoves)
8. else:
9. return None
10. 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)?
11. def getPlayerMove(board):
12. move = ' '
13. while move not in '1 2 3 4 5 6 7 8 9'.split() or not isSpaceFree(board, int(move)):
14. print('What is your next move? (1-9)')
15. move = input()
16. return int(move)

The player records their move with numbers 1-9. The board has numbers labeled from 1-9 and if a space is taken, the game will ask for another number.

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

1. def chooseRandomMoveFromList(board, movesList):
2. possibleMoves = []
3. for i in movesList:
4. if isSpaceFree(board, i):
5. possibleMoves.append(i)
   1. How does the program verify that the space is free?
6. def isSpaceFree(board, move):3
7. return board[move] == ' '

c. What does the program do if there is something wrong with the move?  
74. def chooseRandomMoveFromList(board, movesList):

75. possibleMoves = []

76. for i in movesList:

77. if isSpaceFree(board, i):

78. possibleMoves.append(i)

The program will ask the player to make another move if the previously selected move is already taken. If the player does not input a number from 1-9, the game will ask to input the number again. This is done by repeating the sentence, “What is your next move? (1-9)”.

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?

**Player:**

142. while gameIsPlaying:

143. if turn == 'player':

144. drawBoard(theBoard)

145. move = getPlayerMove(theBoard)

146. makeMove(theBoard, playerLetter, move)

**Computer:**

161. else:

162. move = getComputerMove(theBoard, computerLetter)

163. makeMove(theBoard, computerLetter, move)

The program takes turns between the AI and player.

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

1. def getPlayerMove(board):
2. move = ' '
3. while move not in '1 2 3 4 5 6 7 8 9'.split() or not isSpaceFree(board, int(move)):
4. print('What is your next move? (1-9)')
5. move = input()
6. return int(move)

The program asks “What’s your next move? (1-9)”. It keeps asking until you input a valid move.

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

The program stops asking for moves when someone wins, or when the board is full.

while gameIsPlaying:

if turn == 'player':

drawBoard(theBoard)

move = getPlayerMove(theBoard)

makeMove(theBoard, playerLetter, move)

if isWinner(theBoard, playerLetter):

drawBoard(theBoard)

print('Hooray! You have won the game!')

gameIsPlaying = False

else:

if isBoardFull(theBoard):

drawBoard(theBoard)

print('The game is a tie!')

break

else:

turn = 'computer'

else:

move = getComputerMove(theBoard, computerLetter)

makeMove(theBoard, computerLetter, move)

if isWinner(theBoard, computerLetter):

drawBoard(theBoard)

print('The computer has beaten you! You lose.')

gameIsPlaying = False

else:

if isBoardFull(theBoard):

drawBoard(theBoard)

print('The game is a tie!')

break

else:

turn = 'player'

**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.

The program contains a code in which all the combinations to win are recorded. When one of those combinations is used by either X or O, the game automatically knows that the winning combination has been used and ends the game declaring if you have won or lost.

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

Def checkWinForXorO

if isWinner(theBoard, playerLetter):

drawBoard(theBoard)

print('Hooray! You have won the game!')

gameIsPlaying = False

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

if isWinner(theBoard, playerLetter):

drawBoard(theBoard)

print('Hooray! You have won the game!')

gameIsPlaying = False

1. 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.

The strategy that I use is that If I go first, I place X at #7. Then I place X at #9. After, I place X at 1. The computer will use up its 3 turns blocking me from winning but my fourth move is open for me to win going diagonal at 1,5,9.

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

def isWinner(bo, le):

return ((bo[7] == le and bo[8] == le and bo[9] == le) or

(bo[4] == le and bo[5] == le and bo[6] == le) or

(bo[1] == le and bo[2] == le and bo[3] == le) or

(bo[7] == le and bo[4] == le and bo[1] == le) or

(bo[8] == le and bo[5] == le and bo[2] == le) or

(bo[9] == le and bo[6] == le and bo[3] == le) or

(bo[7] == le and bo[5] == le and bo[3] == le) or

(bo[9] == le and bo[5] == le and bo[1] == le))

1. Modify your program to print a suggested move when it is each player’s turn to move.
2. Demonstrate your AI enhanced game to Mr. Nestor for credit for this level.