**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?")

Type your name comes up and when I typed my name, my name was inserted in the other instruction. It was Hi Muktika how are you?. It looked similar to the following:

Type your name

Muktika

Hi Muktika 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

number = int(input("Enter a number to find square root:"))

if number < 0 :

print ("Please enter a valid number.")

else :

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 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

def tic\_tac\_toe():

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

end = False

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))

def iBoard():

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

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

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

print()

def p1():

n = number()

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

print("You can't go there. Try again")

p1()

else:

board[n] = "X"

def p2():

n = number()

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

print("You can't go there. Try again")

p2()

else:

board[n] = "O"

def number():

while True:

a = input()

try:

a = int(a)

a -= 1

if a in range(0, 9):

return a

else:

print("That's not on the board. Try again")

continue

except ValueError:

print("That's not a number. Try again")

continue

def check\_board():

count = 0

for a in win\_commbinations:

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

print("Player 1 Wins!")

print("Congratulations!")

return True

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

print("Player 2 Wins!")

print("Congratulations!")

return True

for a in range(9):

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

count += 1

if count == 9:

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

return True

while not end:

iBoard()

end = check\_board()

if end == True:

break

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

p1()

print()

iBoard()

end = check\_board()

if end == True:

break

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

p2()

print()

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

print()

tic\_tac\_toe()

tic\_tac\_toe()

1. Provide a complete listing of your program.
   1. Your listing **MUST** include line numbers .
2. 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, else, try, except, for in,
   2. How are moves by player X and player O recorded? If a slot in the grid is already “X” or “O”, the program will inform the player. But if the slot is not, the program will instantly switch the integer in the grid with “X” if it is player 1’s turn or “O” if it is player 2’s turn.

Line 18 or Line 26 [[1]](https://repl.it/@BiaXue/BurdensomeLopsidedPython)

* 1. How are free spaces recorded? As an integer has been inputted or typed when asked to, the integer changes to “X” or “O” depending on the player. However, no changes will occur to the integers that have not been chosen.

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)? By simply, typing in the integer they would like to select.

Line 30 or Line 32[[1]](https://repl.it/@BiaXue/BurdensomeLopsidedPython)

* 1. How does the program verify that the move location is valid? The move is valid as long as the player types in an integer with the range of 0 to 9.

Line 34[[1]](https://repl.it/@BiaXue/BurdensomeLopsidedPython)

* 1. How does the program verify that the space is free? When an integer has been used once, it will immediately switch to “X” or “O” depending on the player. If the grid shows that a number has already been taken through the usage of “X” or”O”, the next time a player inserts an integer already used, the program will tell the player to try again
  2. What does the program do if there is something wrong with the move? If something is wrong with the move, the program will tell the play that the move is not a number and to 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?

while not end:

iBoard()

end = check\_board()

if end == True:

break

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

p1()

print()

iBoard()

end = check\_board()

if end == True:

break

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

p2()

print()

* 1. How does the program keep asking for moves? Unless one of the players does not create a winning combination with on their turn, the program keeps asking for moves.

Line 4[[1]](https://repl.it/@BiaXue/BurdensomeLopsidedPython)

* 1. How does the program decide when to stop asking for moves? If one of the players achieve one of the winning combinations or there are no more moves left, then the program stops asking for moves to either announce for the winner or to restart.

Line 43-60 [[1]](https://repl.it/@BiaXue/BurdensomeLopsidedPython)

**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.