# Final Project

You may choose to do any of the projects below! Feel free to create a game of your own.

Presentation Recap

* Print Statement
  + print(“This is an example”)
  + print(“I am ” + str(30) + “years old.”)
* Declaring and initializing variables
  + variable\_name = “value”
* Booleans:
  + Always True or False
* Comparison Operators:
  + < , <= , > , >=
* Relational Operators:
  + == , !=
* Example of a list

scores = [ 59, 61, 63, 63, 68, 64, 58 ]

Remember that indices start from 0, 1, 2 …

* if condition\_1:

Block of code to be executed

elif condition\_2:

Block of code to be executed

else:

Block of code to be executed

* for item in sequence:

Block of code to be executed

* while condition:

Block of code to be executed

* Mistakes to avoid:

1. Infinite loop—when the condition for your while loop is never false
2. Out of bounds index—make sure that the index you are passing to a list is within bounds (from 0 to 1 less than the length of the list).
3. Working with variables of different data types—typecast variables when necessary
   1. Bad example: string\_sentence = “This is a number: ” + 0
   2. Good example: string\_sentence = “This is a number: ” + str(0)
4. Not returning a value from a function

Online Console: <https://repl.it/languages/python3>

## Project 1 (Text Based Adventure)

In this project, you will build a text based adventure where the player will encounter different “rooms” and can interact with the game by typing responses to prompts.

Requirements:

* Game contains at least 3 “rooms”
* Each “room” must contain a person or object that you can interact with
* Each “room” must connect to one or more other “rooms” (or the exit)
* There’s a way to end the game loop

\*A “room” can be any kind of environment. For example, if you want to set your adventure outdoors, a “room” can be a clearing, or a cave, or a part of the forest.

To build this game you will use your knowledge of if/elif/else statements and conditionals in Python.

### Text-based Adventure Game Pseudocode

|  |
| --- |
| # set up necessary variables  game = true # keeps the while loop running. Set it to false to leave the game loop  room = 1 # keeps track of which “room” user is in. Change to move between “rooms”  while game: # the condition that keeps the game running  # TO DO: if/elif/else statements, each should be a separate room  if room == 1:  # TO DO: print a description about the room and what’s in it  print(“Description goes here”)  # TO DO: offer some options to player  # take in player input  usr\_input = input(“a question goes here”)  # use more if/elif/else statements to between rooms based on user input  if usr\_input == ‘something’: #‘something’ is a condition you set  # TO DO: have user do something. Set room = 2 to move to the next room    elif room == 2: #repeat steps above  elif room == 3: #repeat steps above  # One or more of your rooms must have code that allows you to exit the while loop:  # game = false |

Advanced Challenges:

* Add a mini game or riddle to solve in a room
* Make the adventure a quest where the player needs to complete certain steps before they can finish

## Project 2 (Tic Tac Toe)

In the project, you will build a version of Tic Tac Toe you can play with your friends.

When you are done with this project, your game should be able to:

* Display the gameboard in the console
* Keep track of the player’s moves
* Detect when someone won the game
* Allow players to play again

To build this game you will use your knowledge of lists, conditionals and functions in Python.

### Tic Tac Toe Pseudocode

|  |
| --- |
| def drawBoard(board):  # This function prints out the board that it was passed.  # "board" is a list of 10 strings representing the board (ignore index 0)  print(' | |')  print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])  print(' | |')  print('-----------')  print(' | |')  print(' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])  print(' | |')  print('-----------')  print(' | |')  print(' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])  print(' | |')  def winner(board, player):  # board variable is the list of the gameboard  # player is the player's symbol ("O" or "X")  # TO DO:  # return true if a player has three in a row,  # horizontally, vertically or diagonally  # else, return false  # hint, use boolean operators (“==”) and logical operators (“and”, “or”)  play = True # starts the game loop  while play:  # Reset the board and start game  theBoard = [' '] \* 10  valid\_moves = ['1','2','3','4','5','6','7','8','9']  labels = ['0','1','2','3','4','5','6','7','8','9']  print("Welcome to Tic Tac Toe! Player 1 goes first.")  drawBoard(labels)  # Initialize necessary variables  player = 1  game = True  while game:  # Take in user input  player\_input = input("Input your move: ")  # Check if the move is valid  if player\_input in valid\_moves:  #Mark the board with the necessary mark (“X” or “O”)  if player == 1: # For player 1: O  # TO DO:  # add the move onto the board  # remove player\_input from list of valid\_moves  # change the player number  # check if player has won  if winner(theBoard, "O"):  # TO DO: What happens when a player wins?  # Remember to draw the board!  break # leaves current loop  else:  print('Player 2\'s turn!')  else: # For player 2: X  # TO DO: repeat similar steps from player 1 for player 2  # Add move, remove from list of valid\_moves, check for winner etc.  if winner(theBoard, "X"):  # TO DO: What happens when a player wins?  # Very similar to player 1  break  else:  print('Player 1\'s turn !')  # Print the board to the console  drawBoard(theBoard)  else:  print("Invalid input. Try again")  # End game if someone won or there are not more valid moves (draw)  # After game over, ask if players want to play again  if(len(valid\_moves)==0): #checks if there's a draw, no more moves left  # TO DO: print "Draw" message and end game  if (input("play again? y/n") == "n"):  play = False  print("Thanks for playing!") |

Advanced Challenges:

* Allow players to input and play with unique names (instead of Player 1 and Player 2)
* Make Player 2 automatic

## Project 3 (Battleship)

In this project you will build a simplified, one-player version of the classic board game Battleship! In this version of the game, there will be a single ship hidden in a random location on a 5x5 grid. The player will have 4 guesses to try to sink the ship.

To build this game you will use your knowledge of lists, conditionals and functions in Python.

### Battleship Game Pseudocode

|  |
| --- |
| # import the random functions library  from random import randint  # empty board list  board = []  # the next few lines generate an empty board  for x in range(5):  board.append(["O"] \* 5)  # the next few lines draws the board on the console  def print\_board(board):  for row in board:  print (" ".join(row))  print ("Let's play Battleship!")  print\_board(board)  # the next few lines hide the ship in the board!  def random\_row(board):  return randint(0, len(board) - 1)  def random\_col(board):  return randint(0, len(board) - 1)  ship\_row = random\_row(board)  ship\_col = random\_col(board)  # comment out next two lines while playing as they display the location of the ship  print (ship\_row)  print (ship\_col)  # Everything from here on should go in your for loop!  # Be sure to indent!  for turn in range(4):  guess\_row = int(input("Guess Row:"))  guess\_col = int(input("Guess Col:"))  #TO DO: write condition to test for correct guesses  if (guess is correct):  print ("Congratulations! You sunk my battleship!")  break  else:  #TO DO: write condition for guesses outside the board  if (guess is outside board):  print ("Oops, that's not even in the ocean.")  #TO DO: write condition for repeated guess  elif (already guessed that spot):  print ("You guessed that one already.")  else:  print ("You missed my battleship!")  Set location to “X”  if turn == 3 :  print ("Game Over")  # TO DO:  # write conditional statements to check for the following conditions:  # is the guess correct?  # has the location been guessed already?  # is the guess out of bounds?  # is the guess wrong?  # when the guess is wrong, replace guessed location on the board with an “X”  #NOTE: use the “break” command to exit the loop when the guess is correct  #NOTE: make sure to keep track of the number of turns remaining |

Advanced challenges:

* Make the ship larger than one space, and the player has to hit all the spaces it contains to be able to win.

## Useful functions and libraries

We included some functions and libraries you may find useful when building your project! You can google how to use them or ask one of the instructors.

Functions

* range(*start, stop[,step]*)
  + The range() function returns a list of integers, the sequence of which is defined by the arguments passed to it.
* len(*s*)
  + Return the length (the number of items) of an object. The argument may be a sequence (string, tuple or list) or a mapping (dictionary).
* *str*.lower()
  + Return a copy of the string converted to lowercase
* *str*.upper()
  + Return a copy of the string converted to uppercase
* Break
  + Terminates the loop immediately

Libraries

* Random
* String

Here is a link that you could look up for more information <https://docs.python.org/2/library/functions.html> (Python 2.7)

And the more updated Python 3.0 <https://docs.python.org/3.0/library/functions.html>