* **main()** - of course.
  + Runs the main loop

Keep going set equal to true

If userChoice is 0

Keep going set to false

Elif userChoice gets 1

Print load default game

Elif userChoice gets 2

Print load game file

Elif userChoice 3

Print save current game

Elif userChoice 4

Print edit or create a node

Elif userChoice 5

Print play the current gamehile keep going:

* + Calls a menu

Print starting dictionary with options

Create a variable called decision which gets please choose and then list the 5 options nodes.

Sends control to other parts of the program

* **getMenuChoice()**
  + prints a menu of user options

0) Exit

1. Load default game
2. Load a game file
3. Save the current game
4. Edit or add a node
5. Play the current game
   * repeats if input is invalid

Print What will you do

userChoice = input

If userChoice in (0, 1, 2, 3, 4, 5):

KeepGoing = false

Print Choose something 0-5

* + returns a valid menu choice
* **playGame(game)**

**game is a parameter we will use to pass in the variables**

* + plays the game
  + Keeps going until next node is "quit"
* **playNode()**
  + given the game data and a node,
  + plays out the node
  + returns the next node
* **getDefaultGame()**
  + creates a single-node default game

Lists menu options

Only has a start and quit node

* + returns that data structure

Returns game

* **editNode()**
  + given the current game structure...
  + list all the current node names
  + get a node name
  + if that node exists
    - copy that node to newNode
  + otherwise...
    - create newNode with empty data
  + use editField() to allow user to edit each node
  + return the now edited newNode
* **editField()**
  + get a field name
  + print the field's current value
  + if the user presses 'enter' immediately
    - retain the current value
  + otherwise...
    - use the new value
* **saveGame()**
  + save the game to a data file
  + you can preset the file name (eg 'game.dat')
  + print the current game dictionary in human-readable format
  + Save the file in JSON format
* **loadGame()**
  + presume there is a data file named 'game.dat' in the current directory
  + open that file
  + load the data into the game object
  + return that game object