**FUNCTION:**

* **main()** - of course.
  + Runs the main loop
  + Calls a menu
  + Sends control to other parts of the program
  + Handles invalid input from menu
* **getMenuChoice()**
  + prints a menu of user options
  + returns a menu choice
* **playGame()**
  + 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
  + returns that data structure
* **editNode()**
  + given the current game structure...
  + list all the current node content (json.dumps() would be an easy way to do this)
  + 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

**ALGORITHM:**

**@author: chibuike**

**"""**

import json

def main():

keepGoing = True

game = getDefaultGame()

while (keepGoing):

userChoice = getMenuChoice()

if userChoice == "0":

keepGoing = False

elif userChoice == "1":

game = playGame(game)

#game = getDefaultGame()

# get from getdefaultgame()

elif userChoice == "2":

game = loadGame()

# get from playGame()

elif userChoice == "3":

saveGame(game)

# get from savegame()

elif userChoice == "4":

editNode(game)

# get from editnode()

elif userChoice == "5":

#game = playGame(game)

game = loadGame()

# get from loadgame()

else:

print("chose a number between 0-5 ")

def getMenuChoice():

print ("""

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

""")

userChoice = input("pick an option: ")

return userChoice

def playGame(game):

currentNode = "start"

keepGoing = True

while (keepGoing):

if currentNode == "quit":

keepGoing = False

else:

currentNode = playNode(game, currentNode)

def playNode(game, currentNode):

if currentNode in game.keys():

(description, menuA, nodeA, menuB, nodeB) = game[currentNode]

print(f"""

{description}

1. {menuA}

2. {menuB}

""")

userChoice = input("What will you do? ")

if userChoice == "1":

newNode = nodeA

elif userChoice == "2":

newNode = nodeB

else:

print("Please enter 1 or 2")

newNode = currentNode

return newNode

else:

print("Not a valid Node")

newNode = "quite"

return newNode

def getDefaultGame():

game = {

"start": ["Default start", "start over", "start", "Quit", "quit"]

}

return game

def editNode(game):

allNodes = game.keys()

for node in allNodes:

print(node)

newName = input("please select a node available or name a new node: ")

if newName in allNodes:

newNode = game[newName]

print("Editing node...")

else:

newNode = ("","","","","")

print("creating new node")

(Desc, OptA, gotoA, OptB, gotoB) = newNode

newDesc = editField("description", Desc)

newOptA = editField("option A", OptA)

newgotoA = editField("destination A", gotoA)

newOptB = editField("option B", OptB)

newgotoB = editField("destination B", gotoB)

newNode = (newDesc, newOptA, newgotoA, newOptB, newgotoB)

game[newName] = newNode

return game

def editField(prompt, editing):

field = input(f"input the new {prompt} ({editing}): ")

return field

def saveGame(game):

with open("game.json", "w") as file:

json.dump(game ,file ,indent= 2)

print("Save changes")

print(json.dumps(game ,indent= 2))

def loadGame():

with open("game.json", "r") as file:

game = json.load(file)

print("successfully loaded game")

print(json.dumps(game))

return game

main()