Richard Hayes Crowley

07/2/2021

CSC\_242\_Lab\_08

**SOURCE CODE:**

**Spy\_movie.py**

*import* random

*import* datetime

*from* simple\_term\_menu *import* TerminalMenu

*from* movie\_script *import* MovieScript

*# the movie text and sequences*

seq = []

seq.append("opening teaser sequence")

seq.append("main titles with theme song")

seq.append("the plot unfolds")

seq.append("meeting with the superiors")

seq.append("the gadgets are issued")

seq.append("the mission begins")

seq.append("a romance ensues")

seq.append("thwarted but persistent")

seq.append("physical confrontation with the enemy")

seq.append("the enemy is defeated")

seq.append("the loose ends unfold")

seq.append("on to the next mission")

seq.append("subplot: the drone has launched")

seq.append("subplot: secret vampire society exposed")

seq.append("subplot: Tom Cruise skateboards on an airplane")

*# timings for scenes*

sceneTimings = []

*for* i *in* range(len(seq)-1):

x = random.uniform(0.1, 0.5)

*# using datetime timeDelta package to*

time = str(datetime.timedelta(*hours*=x))[2:-7] + " mins"

sceneTimings.append(time)

*# actor list*

actors = ["Tom Cruise: Vampire", "Tilda Swinton: Vampire", "Chris Rock: Detective", "Angelina Jolie: Archaeologist", "Seth Rogen: Gamer",

"Christopher Walken: Psychic", "Whoopi Goldberg: Nun", "Adam Sandler: Football player", "Jennifer Lopez: Journalist"]

*# searchable list of keywords*

spy\_movie\_keywords = ["drone", "gadgets", "romance", "enemy", "Tom Cruise",

"thwarted", "meeting", "mission", "secret", "theme song"]

def searchKeywords(*ms*):

ls = spy\_movie\_keywords

ls.append("Go Back")

*while* True:

keyword = TerminalMenu(

ls, *title*="\nWhat word would like to search for?").show()

*# "go back" option is last option*

*if* keyword == len(ls) - 1:

*# reset list*

ls = []

*break*

*else*:

search = *ms*.searchKeyword(ls[keyword])

print(search)

def searchActors(*ms*):

ls = actors

ls.append("Go Back")

*while* True:

actor = TerminalMenu(

ls, *title*="\nWhat actor would like to search for?").show()

*# "go back" option is last option*

*if* actor == len(ls) - 1:

*# reset list*

ls = []

*break*

*else*:

search = *ms*.searchActors(ls[actor])

print(search)

def main():

*# create a new movie script object*

ms = MovieScript(seq, actors, sceneTimings)

print("\n~~~~~ Welcome to the spy movie app! ~~~~~~")

*while* True:

entry\_point = TerminalMenu(

["View Script", "View Call Sheet", "View Scene Timings", "Search Keyword(s)", "Search Actors", "Shuffle Script", "Exit"], *title*="\nWhat would you like to do?").show()

*if* entry\_point == 0:

*# calling \_\_str\_\_ method on MovieScript class*

print("\n")

print(ms)

*elif* entry\_point == 1:

print("\n")

print(ms.printCallSheet())

*elif* entry\_point == 2:

print("\n")

print(ms.printSceneTimings())

*elif* entry\_point == 3:

print("\n")

searchKeywords(ms)

*elif* entry\_point == 4:

print("\n")

searchActors(ms)

*elif* entry\_point == 5:

print("\n")

shuffledScript = seq

random.shuffle(shuffledScript)

ms = MovieScript(shuffledScript, actors, sceneTimings)

print(ms)

*else*:

print("\n")

print("Goodbye!")

exit()

*if* \_\_name\_\_ == "\_\_main\_\_":

main()

**MovieScript.py**

*import* random

class MovieScript(object):

def \_\_init\_\_(*self*, *script*, *actors*, *sceneTimings*) -> None:

*self*.script = *script*

*self*.actors = *actors*

*self*.sceneTimings = *sceneTimings*

*# convert the sequence list to a dictionary, with random actors as values*

*self*.callSheet = {idx: {"sequence": x, "actors": random.sample(

*self*.actors, 3)} *for* idx, x *in* enumerate(*self*.script)}

def \_\_iter\_\_(*self*):

cursor = 0

*while* cursor < len(*self*.script):

*# yield sends each item to the caller, the for loop*

*yield* *self*.script[cursor]

cursor += 1

def \_\_str\_\_(*self*):

str = ""

*for* x *in* range(len(*self*.script)):

str += f"\u2022 {*self*.script[x]} \n"

*return* str

*# accessors*

def getScript(*self*) -> list:

*return* *self*.script

def getActors(*self*) -> list:

*return* *self*.actors

def getCallSheet(*self*) -> dict:

*return* *self*.callSheet

def getSceneTimings(*self*) -> list:

*return* *self*.sceneTimings

def printCallSheet(*self*) -> str:

*return* "\n".join([(f"Sequence {i + 1}: {x['sequence']}, Actors: {', '.join([a *for* a *in* x['actors']])}") *for* (i, x) *in* *self*.callSheet.items()])

def printSceneTimings(*self*) -> str:

*return* "\n".join([f'Scene {i}: {*self*.sceneTimings[i]}' *for* i *in* range(len(*self*.script) - 1)])

def searchKeyword(*self*, *keyword*) -> str:

str = ""

*for* i *in* range(len(*self*.script)):

*if* (*keyword* in *self*.script[i]):

str += f"\n{*keyword*} found in scene sequence number {i + 1}: \"{*self*.script[i]}\""

*return* str *if* len(str) > 0 *else* f"{*keyword*} not found!"

def searchActors(*self*, *actor*) -> str:

str = ""

*for* key *in* *self*.callSheet.keys():

*if* (*actor* in *self*.callSheet[key]['actors']):

str += f"\n{*actor*} found in scene sequence number {key}\nSequence: \"{*self*.callSheet[key]['sequence']}\" \nActor(s): {[f'{i} ' *for* i *in* *self*.callSheet[key]['actors']]}"

*return* str *if* len(str) > 0 *else* f"{*actor*} not found!"

*# mutators*

def setScript(*self*, *s*) -> None:

*self*.script = *s*

**OUTPUT:  
  
See attached video for demo!**