Joshua Finlayson

SN: 10691485

Main Code

Initialise global constant *DATA_FILE_PATH* = “data.txt”

Initialise *data* to be an empty list

Open the *DATA_FILE_PATH* file in read mode as *file*

Append all the JSON data in *file* as multiple different dictionaries onto the end of *data*

If the file directory can’t be accessed

    Print “The directory ” + *DATA_FILE_PATH* + “ is invalid, or can’t be reached”

    Close the program

If the file was empty

    Print “The file ” + *DATA_FILE_PATH* + “ was empty”

Close *file*

Print “Welcome to Joshua's Boardgame Catalogue Admin Program.”

Loop indefinitely

    Print “Choose [a]dd, [l]ist, [s]earch, [v]iew, [d]elete or [q]uit.”

    Print "For (s)earch, (v)iew, and (d)elete if you type the specifier after the letter the command runs. (e.g. 's Hello' searches for the 'Hello' term)"

    Get user input and save as *inp*

Set *inp* to *inp* in all lowercase

Strip all leading/trailing whitespace from *inp*

If the first word in *inp* is “a”

    Initialise *new_data* as an empty dictionary

    Set *new_data*[‘name’] to the return of **input_string** passing in “Enter boardgame name: ”

    Set *new_data*[‘year’] to the return of **input_int** passing in “Enter release year: ”, 0, and the current year

    Initialise bool *already_in* as false

    For *d* in *data*

        If *d*[‘name’] in lowercase equals *new_data*[‘name’] in lowercase and *d*[‘year’] equals *new_data*[‘year’]

            Set *already_in* to true

    If *already_in* is true

        Print “That boardgame is already in the database”

        Restart the loop

    Set *new_data*[‘desc’] to the return of **input_string** passing in “Enter short description: ”

    Set *new_data*[‘players’] to the return of **input_range** passing in “Enter number of players as a range e.g. 1-4: ”

    Set *new_data*[‘playtime] to the return of **input_range** passing in “Enter playtime in minutes as a range e.g. 15-30: ”

    Set *new_data*[‘min_age’] to the return of **input_int** passing in “Enter the minimum recommended playing age: ”, and 0

    Set *new_data*[‘complexity’] to the return of **input_int** passing in “Enter complexity(1-5): ”, 1, and 5

    Append *new_data* to *data*

Print *new_data*[‘name’] + “ added”

Call **save_data** passing in *data*

Else if the first word in *inp* is “l”

If *data* is empty

Print “No boardgames saved”

Restart the loop

Print “List of Boardgames”

Loop for the length of *data* saving the enumeration number as *i*

Print “ ” + *i* + 1 + “) ” + *data*[*i*][‘name’] + “ (” + *data*[*i*][‘year’] + “)”

Else if the first word in *inp* is “s”

If *data* is empty

Print “No boardgames saved”

Restart the loop

Initialise *search_term* to be an empty string

If *inp* was more than one word:

Set *search_term* to be *inp* without the first word

Else:

Set *search_term* to the return of **input_something** passing in “Enter a search term: ”

Set *search_term* to *search_term* in all lowercase

Initialise *search_results* as an empty list

Loop for the length of *data* saving the enumeration number as *i*

If *data*[*i*][‘name’] or *data*[*i*][‘desc’] contains *search_term*

Initialise *lst* to a lost containing *i* and *data*[*i*]

Append *lst* to the end of *search_results*

If *search_results* is an empty list

Print “No results found”

Restart the loop

Print “Search results: ”

Loop for the length of *search_results* saving the enumeration number as *i*

Print (*search_results*[*i*][0] + 1 + “) ” + *search_results*[*i*][1][‘name’] + “ (” + *search_results*[*i*][1][‘year’] + “)”

Else if the first word in *inp* is “v”

If data is empty

Print “No boardgames saved”

Restart the loop

Set *num* to the return of **single_input_parsing** passing in *data*, *inp*, and “Boardgame number to view: ”

Print *data*[*num*][‘name’] + “ (” + *data*[*num*][‘year’] + “)”

Print *data*[*num*][‘desc’]

Print “ Players: ” + *data*[*num*][‘players’][0] + “-” + *data*[*num*][‘players’][1]

Print “ Playtime: ” + *data*[*num*][‘playtime’][0] + “-” + *data*[*num*][‘playtime’][1] + “ minutes”

Print “ Age: ” + *data*[*num*][‘min_age’] + “+”

Print " Complexity: " + *data*[*num*]['complexity'] + "/5"

Else if the first word in *inp* is "d"

If *data* is empty

Print "No boardgames saved"

Restart the loop

Set *num* to the return of **single_input_parsing** passing in *data*, *inp*, and "Boardgame number to delete: "

Delete the item at index *num* from *data*

Print "Deleted boardgame"

Call **save_data** passing in *data*

Else if the first word in *inp* is "q"

Print "Goodbye"

Print "Press enter to close the program"

Wait till the user presses enter

Break out of the loop

Else

Print "Invalid Choice. Please try again."

Functions

Function **input_something** with parameter *prompt*

Loop forever

Print *prompt*

Get user input and save is as *inp*

Set *inp* to *inp* with all whitespace from the end removed

If *inp* contains any value

Return *inp*

Print “Sorry, you didn’t seem to input anything there. Please try again”

Function **input_string** with parameter *prompt*

Loop Endlessly

Set *inp* to the return of **input_something** passing in *prompt*

If *inp* is a number

Print "Please input a string, not a number"

Else

return *inp*

Function **input_int** with parameters *prompt*, *min_value*¸ *max_value*

Loop forever

Call **input_something** passing in *prompt*

Save the return as *inp*

If *inp* is not an integer

Print “You need to input an integer (a whole number)”

Restart the loop

Set *inp* to be an integer

If *min_value* had a value passed into the function

If *inp* is less than *min_value*

Print “The input cannot be less than ” + *min_value*

Restart the loop

If *max_value* had a value passed into the function

If *inp* is greater than *max_value*

Print “The input cannot be more than ” + *max_value*

Restart the loop

Return *inp*

Function **input_range** with parameter *prompt*

Loop forever

Print *prompt*

Get user input and save as *inp*

Set *inp* to be *inp* with all whitespace removed

If *inp* does not have a ‘-’ somewhere in it

Print “You need to input a range separated with a ‘-’”

Restart the loop

If *inp* has more than one ‘-’ in it

Print “You can only have one ‘-’, so no negative numbers”

Restart the loop

Split *inp* up into two numbers along the ‘-’ and save them as *num1* and *num2*

If either *num1* or *num2* are not integers

Print “You must input two integers on either side of the ‘-’”

Restart the loop

If either *num1* or *num2* are less than or equal to 0

Print “You must input positive numbers greater than zero”

Restart the loop

If *num2* is less than *num1*

Print “The second number must be greater than the first number”

Restart the loop

Return list of two elements: *num1*, and *num2*

Function **save_data** with parameter *data*

Open the *DATA_FILE_PATH* file in write mode as *file*

Overwrite anything already within *file* with *data*

Close *file*

Function **single_input_parsing** with parameters *data*, *inp*, and *incorrect_prompt*

Initialise int *num* as -1

If *inp* is more than one word:

Initialise *str_num* to *inp* without the first word

if *str_num* can be converted into an integer and *str_num* is between 1 and the length of *data*

num = *str_num* as an integer – 1

else:

Print *str_num* + " is not a valid input. Please try again"

if *num* is -1:

Set *num* to the return of **input_int** passing in *incorrect_prompt*, 1, and the length of *data*

Set *num* to *num* - 1