codecademy

Jeopardy!
The Console Game

Pandas Python library

Alex Ricciardi

Date: 05/24/2020

Project Overview

The Jeopardy Console Game is a personnel project, an offshoot from the Codecademy Data Science path, under the Data Analysis with Pandas Phyton library section.

It is a Python Pandas library based, solo player Jeopardy console game.

The project features:

- Tidying data from a csv file to be used by the console game.
- Data manipulation with the python Pandas library.
- User input with msvcrt.getch().
- Error handling.

Game play description:

The Gameplay consists of a questions/clues quiz comprising of 3 rounds.

The clues in the quiz are presented as "answers" and responses must be phrased in the form of a question..

- Round-1 Jeopardy: 2 categories 2 clues.
- Round-2 Double Jeopardy: 2 categories 2 clues, the clues values are double.
- Round-3 Final Jeopardy: 1 clue wager.

- To move from round-1 to round-2 and from round- to round-3, all the clues in all the categories in the round, have to be answered, and your winnings can not be $0 or less.

- The game feature a settings options where the number of categories and the number of clues per category can be change up to 4 categories and 4 clues per category.

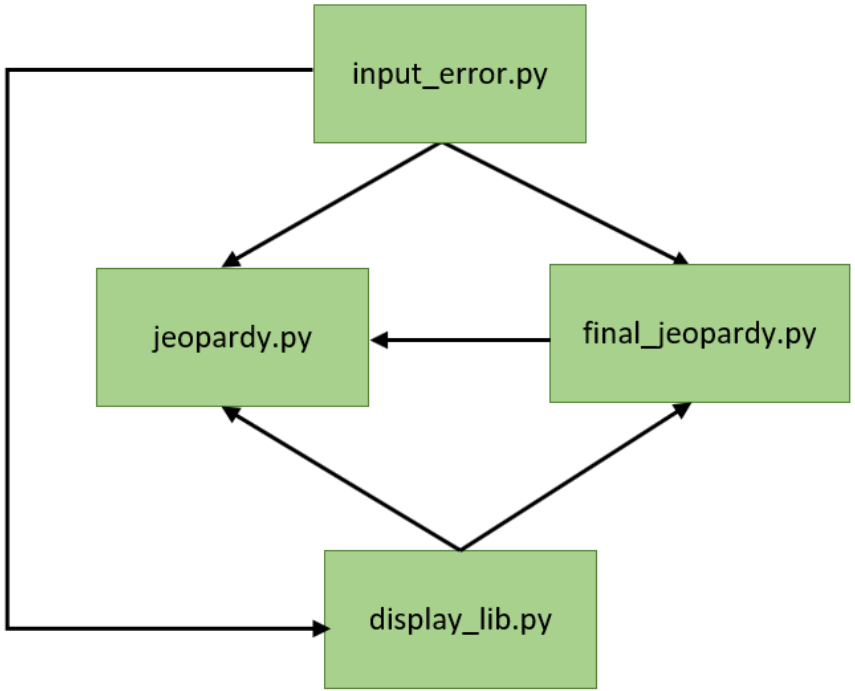
- The game also feature a cheat mode when activated will display the response to the clues.

Related links:

Code Sections :

Code section flow chart representing the relationship between the game code sections.

The sections are saved in four different python files, with the jeopardy.py been the game code main-section, and final_jeoprady.py, display.py and input_error.py been the game code sub-sections.

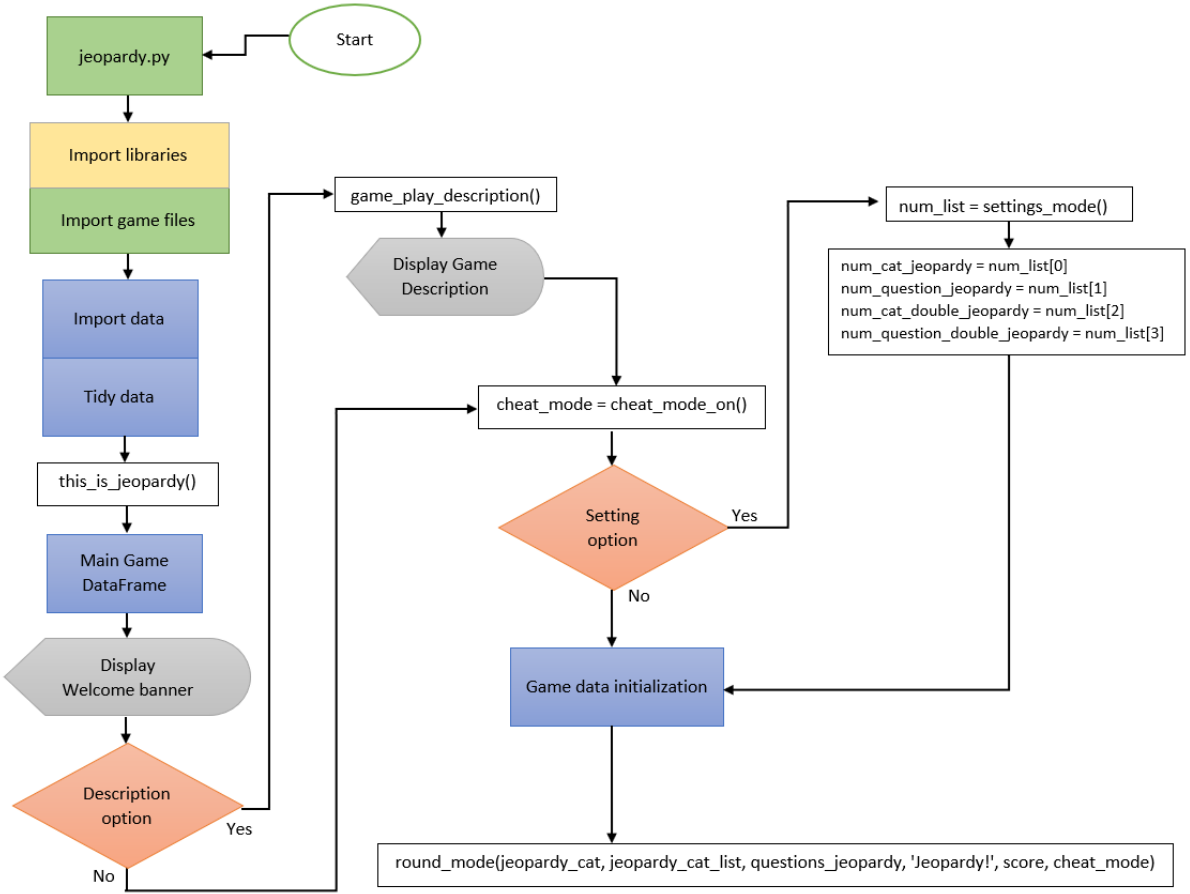


Section-1:
jeopardy.py - Game Initialization

1.1 overview

- Import libraries
- Import game files
- Import data
- Tidy data
- Launch This Is Jeopardy
- Description option
- Cheat mode
- Settings option
- Game data initialization
- Launch Round-1Jeopardy!

1.2 Game Initialization Flowchart



1.3 Importing Libraries

Code lines: 20-25 jeopardy.py

```
import pandas as pd
import numpy as np
# getch() for windows operating system
import msvcrt
```

msvcrt provides access to some useful capabilities on Windows platforms, in this project the following functionality for Console I/O were used:

- msvcrt.getch()
  Read a keypress and return the resulting character as a byte string. Nothing is echoed to the console. This call will block if a keypress is not already available but will not wait for Enter to be pressed.

For more information about msvcrt:
https://docs.python.org/3/library/msvcrt.html

If you are interested to make the Console I/O character capturing compatible on multiple platforms:
https://docs.python.org/3/library/curses.html#curses.window.getch
https://docs.python.org/3/howto/curses.html#curses-howto

Note:
If you are using the IDE Pycharm's console, msvcrt.getch() will not work, you need to use the window command prompt console.

1.4 Importing Game Files

The fowling functions , from the game code sub-sections, are utilized by jeopardy.py.

input_error.py **functions:**

- check_num_error()
- check_error_question()
- check_error_cat()

display_lib.py **functions:**

- display_categories()
- questions_display()
- question_selected_display()
- quit_game()
- wipe_screen()

final_jeopardy.py **function**:

- final_jeopardy_round()

Code lines: 20-23 jeopardy.py

```
# Input errors handling functions
import input_error as ie
# Display functions for the Jeopardy!
# and Double Jeopardy Rounds
import display_lib as dl
# Final Jeopardy! Round functions
import final_jeopardy as fj
```

1.5 Import Data

The data is made available by Codecademy Data Science path, under the Data Analysis with Pandas Phyton library section.
The data is available as a CSV (comma-separated values) format, and it is base on the show numbers and air dates of the televised Jeopardy game show.

https://www.codecademy.com/practice/projects/this-is-jeopardy

Code line: 77 jeopardy.py

```
#-----------------------Importing jeopardy data files
jeopardy = pd.read_csv("data/jeopardy.csv")
```

Data Table Sample

index	Show Number	Air Date	Round	Category	Value	Question	Answer
0	4680	12/31/2004	Jeopardy!	HISTORY	$200	For the last 8 years of his life, Galileo was under house arrest for espousing this man's theory	Copernicus
1	4680	12/31/2004	Jeopardy!	ESPN's TOP 10 ALL-TIME ATHLETES	$200	No. 2: 1912 Olympian; football star at Carlisle Indian School; 6 MLB seasons with the Reds, Giants & Braves	Jim Thorpe
2	4680	12/31/2004	Jeopardy!	EVERYBODY TALKS ABOUT IT...	$200	The city of Yuma in this state has a record average of 4,055 hours of sunshine each year	Arizona
3	4680	12/31/2004	Jeopardy!	THE COMPANY LINE	$200	In 1963, live on "The Art Linkletter Show", this company served its billionth burger	McDonald's
5	4680	12/31/2004	Jeopardy!	EPITAPHS & TRIBUTES	$200	Signer of the Dec. of Indep., framer of the Constitution of Mass., second President of the United States	John Adams

1.6 Data Tidying

The original data needs some tidying before it can be used to build the game data itself.
For example:
The lambda function in jeopardy = jeopardy.rename() removes the unwanted whitespace at the beginning of the column names and keep the column name 'Show Number' unchanged.

Code lines: 81-96 jeopardy.py

```
# Renaming the mis-formatted columns names
jeopardy = jeopardy.rename(columns=lambda column_name: column_name[1:] if column_name != 'Show Number' else column_name)
# Re-formatting rows with a NaN Answer value
jeopardy = jeopardy.fillna(value={'Answer' : 'Null'})
# Reformatting the column (Value) by removing ($) and (,) turning the data value into integers
jeopardy.Value = jeopardy['Value'].replace('[\$,]', '', regex=True)
# Replacing the (None) values with (0)
jeopardy.Value = jeopardy['Value'].replace('None', '0')
# Value column from string type to an integer type
jeopardy.Value = pd.to_numeric(jeopardy.Value)
# Removing Hyperlinks
jeopardy.Question = jeopardy['Question'].replace('(\(?<.*>\.?\)?)', '', regex=True)
# Removing spaces
jeopardy.Question = jeopardy['Question'].replace(r'^\s*$', np.nan, regex=True)
# Removing [video clue]
jeopardy.Question = jeopardy['Question'].replace(r'[\[video clue\]]', np.nan, regex=True)
```

1.7 Game Section launch and Variables

The function this_is_jeopardy() when call launches the game section.

The fallowing variables set the number of categories and the number of questions per category for the Jeopardy! And Double Jeopardy! rounds.

- num_cat_jeopardy
- num_question_jeopardy
- num_cat_double_jeopardy
- num_question_double_jeopardy

The score variable keeps track of the player winnings.

Code line:725 jeopardy.py

```
this_is_jeopardy()
```

Code lines: 112-115 jeopardy.py

```
# --------------------------
# ----------- Variables
# --------------------------
#
# Settings variable
num_cat_jeopardy = 2
num_question_jeopardy = 2
num_cat_double_jeopardy = 2
num_question_double_jeopardy = 2
#
# Winnings tracker
score = 0
```

1.8.0 Main Game DataFrame, tidying the data

Creating a Pandas DataFrame that is more suited to the game needs.

Tidying the data by removing unusable data.

Code lines: 340-355 jeopardy.py

```
# Selecting only the needed columns
df_jeopardy_game = df_jeopardy_game.loc[:, ['Category', 'Value', 'Question', 'Answer']]
# Reformatting Question values made of only spaces with the value NaN
df_jeopardy_game.Question = df_jeopardy_game.Question.replace(r'^\s*$', np.nan, regex=True)
# Reformatting Question values of [video clue] with the value NaN
df_jeopardy_game.Question = df_jeopardy_game.Question.replace(r'[\[video clue\]]', np.nan, regex=True)
# Removing parenthesis from the Answer column values
df_jeopardy_game.Answer = df_jeopardy_game.Answer.replace(r'\(|\)', '', regex=True)
# Removing rows with the category VWLLSS FRT
df_jeopardy_game = jeopardy.loc[jeopardy.Category != 'VWLLSS FRT']
# Removing rows with the category VWLLSS FLWRS
df_jeopardy_game = df_jeopardy_game.loc[jeopardy.Category != 'VWLLSS FLWRS']
# Removing rows with the category VWLLSS VGTBLS
df_jeopardy_game = df_jeopardy_game.loc[jeopardy.Category != 'VWLLSS VGTBLS']
# Removing rows with the category VWLLSS CNTRS
df_jeopardy_game = df_jeopardy_game.loc[jeopardy.Category != 'VWLLSS CNTRS']
# Removing empty questions with a values NaN
df_jeopardy_game = df_jeopardy_game.loc[df_jeopardy_game.Question.notna()].reset_index(drop=True)
```

Reformatted Data Table Sample

index	Category	Value	Question	Answer
0	HISTORY	$200	For the last 8 years of his life, Galileo was under house arrest for espousing this man's theory	Copernicus
1	ESPN's TOP 10 ALL-TIME ATHLETES	$200	No. 2: 1912 Olympian; football star at Carlisle Indian School; 6 MLB seasons with the Reds, Giants & Braves	Jim Thorpe
2	EVERYBODY TALKS ABOUT IT...	$200	The city of Yuma in this state has a record average of 4,055 hours of sunshine each year	Arizona
3	THE COMPANY LINE	$200	In 1963, live on "The Art Linkletter Show", this company served its billionth burger	McDonald's
5	EPITAPHS & TRIBUTES	$200	Signer of the Dec. of Indep., framer of the Constitution of Mass., second President of the United States	John Adams

1.8.1 Removing categories with less than 4 questions

The data came with an extensive amount of categories having less than 4 questions.

The fallowing code removes categories with less than 4 questions

Code lines: 359-370 jeopardy.py

```
# ---- Removing Duplicate and categories  with less than 3 questions
#
# Creating a category DataFrame with the number of unique questions per category
category_q_count = df_jeopardy_game.Category.value_counts()
category_q_count = category_q_count.reset_index() # pd.DataFrame type
# Renaming columns, index named column can create issues
category_q_count = category_q_count.rename(columns = {"index" : "category", "Category" : "num_of_questions"})
# Categories with 3 or less of questions
category_3q_count = category_q_count.loc[category_q_count.num_of_questions <= 3, 'category'] # pd.Series type
# From a pd.Series type to an object list type
category_3q_count = list(category_3q_count)
# Removing categories with 3 or less questions
for category in category_3q_count:
    df_jeopardy_game = df_jeopardy_game.loc[df_jeopardy_game.Category.apply(not_jeopardy_category)]
```

1.9 Welcome Banner and wipe_screen() function

The triple single quotes (''' ''') creates strings that span multiple lines.
The if statement in the os.system() function checks the operating system name and clears the console display.

The dl.wipe_screen() function wipes the console screen clear, the function is part of the game code sub-section display_lib.py.

Code lines: 564-573 jeopardy.py

```
# Wipe console display
dl.wipe_screen()
#
print('''
              ********************************************************************************************************************************
              ********************************************************************************************************************************
              ***                                                                                                                          ***
              ***                                        WELCOME TO THIS IS JEOPARDY!                                                      ***
              ***                                                                                                                          ***
              ********************************************************************************************************************************
              ********************************************************************************************************************************''')
```

1.10 Description Option

Note :
that msvcrt.getch() from the library msvcrt is utilized to capture the user character input, the b in b'd' stands for byte base data.

Code lines: 575-588 jeopardy.py

```
print('''
                                ********************************************************
                                *                                                      *
                                *        Press D for the game play description         *
                                *        or any other key to continue                  *
                                *                                                      *
                                ********************************************************''')
#
# ----------------------------------------
# ----------- Enter Description option?
# ----------------------------------------
#
# Checks for user choice and redirects if need it
desp = msvcrt.getch()
if desp.lower() == b'd': game_play_description()
```

Code lines: 251-331 jeopardy.py

```
def game_play_description():
     print('''
**************************************************************************************************************
**                                                                                                          **
**   "This is Jeopardy"                                                                                     **
**                                                                                                          **
**     The game play consists of a clue quiz comprising of 3 rounds.                                        **
**                                                                                                          **
**     The clues in the quiz are presented as "answers",                                                    **
**     and responses must be phrased in the form of a question.                                             **
**                                                                                                          **
**                                                                                                          **
**************************************************************************************************************
……..
```

1.11 Cheat mode

If cheat mode is enable, cheat_mode = True, the question_selected_display() function, from display_lib.py, will display on the console the clue answers.

Cheat mode is a great tool to troubleshoot the functionality of the game code.

Code line: 593 jeopardy.py

```
cheat_mode = cheat_mode_on()
```

Code lines: 214-240 jeopardy.py

```
def cheat_mode_on():
    # Cheat mode tracker
    cheat_mode = False
    print('''
              ************************************************************

               Do you want enable cheat mode?

               In cheat mode,
               the answers to the clues will be displayed on the screen

              ************************************************************

                    Press Y for yes and N for No

              ************************************************************''')
    key = msvcrt.getch()
    if key.lower() == b'y':
        cheat_mode = True
        return cheat_mode
    elif key.lower() == b'n':
        return cheat_mode
    # User enter an invalid input
    else:
        print('---------------------- error\n wrong key.........')
        # calls back
        cheat_mode_on()
```

1.12.0 Do you want enter settings option?

The user can choose to enter the settings option and change the number of categories and the number of questions per category for the Jeopardy! and Double Jeopardy! Rounds.
If the user chooses to not enter the settings option the numbers will stay equal to their set values of 2, see the slide 1.5 Variables.

The settings_option() function when call will display a console interface where the user can change the numbers.
The function also returns the changed numbers in a list, num_list, and the numbers are applied to the variables:

- num_cat_jeopardy
- num_question_jeopardy
- num_cat_double_jeopardy
- num_question_double_jeopardy

Code lines: 599-620 jeopardy.py

```
print('''
*******************************************************

 Do you want enter settings option?

 In the settings mode,
 you can change
 the numbers of categories and clues per category

 *******************************************************

 Press Y for yes or any another key to continue

 *******************************************************
''')
    key = msvcrt.getch()
    #
    if key.lower() == b'y':
        num_list = settings_option()
        num_cat_jeopardy = num_list[0]
        num_question_jeopardy = num_list[1]
        num_cat_double_jeopardy = num_list[2]
        num_question_double_jeopardy = num_list[3]
```

1.12.1 Settings Option Function

The settings_option() function creates a 4 rows num_list with values set to 2,
then calls the function num_settings() function to store the user inputted values and returns num_list.

Code lines: 175-208 jeopardy.py

```
    def settings_option():
        # Stores user numbers choices
        num_list = [2, 2, 2, 2, 2]
        print('''
**************************************************************
* Enter the number of categories for the Jeopardy! Round, up to 4 *
**************************************************************''')
        num_list[0] = num_settings()
        print('''
**************************************************************************
* Enter the number of clues per categories for the Jeopardy! Round, up to 4 *
**************************************************************************''')
        num_list[1] = num_settings()
        print('''
**************************************************************************
* Enter the number of categories for the Double Jeopardy! Round, up to 4 *
**************************************************************************''')
        num_list[2] = num_settings()
        print('''
********************************************************************************
* Enter the number of clues per categories for the Double Jeopardy! Round, up to 4 *
********************************************************************************''')
        num_list[3] = num_settings()
        print('''
*************************
* Press a key to continue *
*************************''')
```

1.12.2 Number Settings Function

The num_settings() is the input and error handling function for the settings option module. The function returns num, the number inputted by the user.

The ie.check_num_error() function checks if a character type was entered instead of a number type, the function is part of the game code sub-section input_error.py.

Note: .decode('utf-8') prevent the 'b', byte type, to be displayed.
For example:
num = msvcrt.getch()
print(num)
>> b'4'
Print(num.decode('utf-8'))
>> 4

Code lines: 147-171 jeopardy.py

```
def num_settings():
    num = msvcrt.getch()
    print('\n You entered: ' + num.decode('utf-8'))
    print()
    # Checks if a no-numeric key was pressed by user
    while ie.check_num_error(num):
        print('\n-------------------------------- Error')
        print(' Wrong entry\n')
        print('-------------------------------- Error')
        print('\n************************\n Please a number between 1 and 4')
        num = msvcrt.getch()
        print(' You entered: ' + num.decode('utf-8'))
        print()
    # From byte type to integer type
    num = int(num)
    # Checks if an invalid numeric key was pressed by user
    if num == 0 or num > 4:
        print('\n-------------------------------- Error')
        print(' Wrong entry\n')
        print('-------------------------------- Error')
        print('\n************************\n Please a number between 1 and 4')
        # Calls Back
        num = num_settings()
    # Returns the number as integer 0<>4
    return num
```

1.13.0 Game data initialization categories round-3 and round-2

I utilize the .sample() Pandas function with n=1 to generate a random sample row, to select a category/question for the Final Jeopardy! Round
I utilize the function sample_dupl() and the variable num_cat_double_jeopardy to select the categories for the Double Jeopardy! round

Code lines: 635-652 jeopardy.py

```
# ----------------------------------------
# ---------- Categories initialization
# ----------------------------------------
# ------ Final Jeopardy! Round-3
# Randomly selecting a category/question
final_jeopardy = df_jeopardy_game.sample(n=1).reset_index(drop=True)  # pd.Series Type
final_jeopardy.cat = str(list(final_jeopardy.Category))
#
# ------ Double Jeopardy! Round-2
#
# Creating a jeopardy_game DataFrame without the category from round-3
double_jeopardy_game = df_jeopardy_game.loc[df_jeopardy_game.Category != final_jeopardy.cat]
# Sorting the Double Jeopardy round DataFrame by Category and value columns
double_jeopardy_game = double_jeopardy_game.sort_values(by=['Category', 'Value']).reset_index(drop=True)
# Randomly selecting categories and check for duplicates
double_jeopardy_cat = sample_dupl(double_jeopardy_game.Category, num_cat_double_jeopardy) # pd.Series Type
# From a pd.Series type to an object list type to be used in a for loop
double_jeopardy_cat_list = list(double_jeopardy_cat)  # list type
# double_jeopardy_cat to pd.DataFrame type
# (note: using reset_index(drop=True) will keep double_jeopardy_cat as a pd.Series)
double_jeopardy_cat = double_jeopardy_cat.reset_index()
# Adding a Boolean column to track if all the question in a category have been answered
double_jeopardy_cat['questions_not_answered'] = True
……..
```

1.13.1 The sample_dupl(df, num) function

The sample_dupl(df, num) is utilized to randomly select rows from a DataFrame,
it takes two parameters a DataFrame, df, and the number of rows, num.
sample_dupl() utilizes the Pandas function .sample() to generate a random sample of rows, and the Pandas function .duplicated() to check for duplicated rows.

I utilize sample_dupl() in the game data initialization section of jeopardy.py, to select categories and questions to be use in the Jeopardy! and Double Jeopardy! Rounds, the function returns a Pandas Series.

Code lines: 133-139 jeopardy.py

```
def sample_dupl(df, num):
    series_d = df.sample(n=num) # pd.Series
    # Checks for duplicates
    series_duplicates = series_d.duplicated()
    if series_duplicates.any():
        sample_dupl(df, num)
    return series_d
```

1.13.2 Game data initialization categories round-1

I utilize the function sample_dupl() and the variable num_cat_jeopardy to select the categories for the Jeopardy! Round and .apply() the lambda function not_jeopardy_category to double_jeopardy_game to create a Jeopardy! Round DataFrame without the categories already selected for the Double Jeopardy! Round.

Code lines: 657-670 jeopardy.py

```
# ------ Jeopardy! Round-1
#
# Creating a jeopardy_game DataFrame without the categories from round-3 and 4
jeopardy_game = pd.DataFrame()
for category in double_jeopardy_cat_list:
    jeopardy_game = double_jeopardy_game.loc[double_jeopardy_game.Category.apply(not_jeopardy_category)]
# Sorting the Jeopardy round DataFrame by Category and value columns
jeopardy_game = jeopardy_game.sort_values(by=['Category', 'Value']).reset_index(drop=True)
# Randomly selects categories and checks for duplicated
jeopardy_cat = sample_dupl(jeopardy_game.Category, num_cat_jeopardy)  # pd.Series Type
# From a pd.Series type to an object list type to be used in a for loop
jeopardy_cat_list = list(jeopardy_cat)  # list type
# jeopardy_cat to pd.DataFrame type
# (note: using reset_index(drop=True) will keep jeopardy_cat as a pd.Series)
jeopardy_cat = jeopardy_cat.reset_index()  # pd.DataFrame type
# Adding a Boolean column to track if all the question in a category have been answered
jeopardy_cat['questions_not_answered'] = True
```

Code line: 132 jeopardy.py

```
# The lambda function is Not-selecting data by a given category
not_jeopardy_category = lambda x: x != category
```

1.14.0 Game data initialization questions Double Jeopardy! round

I utilize the function sample_dupl(), the variable num_question_double_jeopardy and the lambda function jeopardy_category to select the questions from the Double Jeopardy! Round selected categories.
Note: that in the Double Jeopardy! Round the question values are doubled.

Code lines: 693-710 jeopardy.py

```
# ------------------------------------------------------------
# ----------- Questions for each category initialization
# ------------------------------------------------------------
# ------ Double Jeopardy! Round-2
# Creates a jeopardy_game DataFrame containing
# all the randomly selected Double Jeopardy rows questions
questions_double_jeopardy = pd.DataFrame()
for category in double_jeopardy_cat_list:
    # DataFrame storing all the rows for the randomly selected categories
    double_jeopardy_cat_rows = double_jeopardy_game.loc[double_jeopardy_game.Category.apply(jeopardy_category)]
    # Randomly selects rows (question) for each randomly selected categories and checks for duplicated
    double_jeopardy_rows = sample_dupl(double_jeopardy_cat_rows, num_question_double_jeopardy) # pd.Series Type
    # DataFrame storing the randomly selected rows (question) for each randomly selected categories
    questions_double_jeopardy = questions_double_jeopardy.append(double_jeopardy_rows, ignore_index=True)
# Adding the not-answered column and set to true
questions_double_jeopardy['not_answered'] = True
# In the Double Jeopardy round the question values are Doubled
questions_double_jeopardy.Value = questions_double_jeopardy.Value.apply(lambda x: x * 2)
# Sorting the Double Jeopardy round DataFrame by Category and value columns
questions_double_jeopardy = questions_double_jeopardy.sort_values(by=['Category', 'Value']).reset_index(
    drop=True)
```

Code lines: 145 jeopardy.py

```
# The lambda function is selecting data by a given category
jeopardy_category = lambda x: x == category
```

1.14.1 Game data initialization questions Jeopardy! round

I utilize the function sample_dupl(), the variable num_question_double_jeopardy and the lambda function jeopardy_category to select the questions from the Jeopardy! Round selected categories.

The round_mode() function launches the Jeopardy! Round.

Code lines: 717-729 jeopardy.py

```
# ------ Jeopardy! Round-1
#
# Creates a jeopardy_game DataFrame containing
# all the randomly selected jeopardy rows questions
questions_jeopardy = pd.DataFrame()
questions_jeopardy.name = 'questions_jeopardy'
for category in jeopardy_cat_list:
    # DataFrame storing all the rows for the randomly selected categories
    jeopardy_cat_rows = jeopardy_game.loc[jeopardy_game.Category.apply(jeopardy_category)]
    # Randomly selects rows (question) for each randomly selected categories and checks for duplicated
    jeopardy_rows = sample_dupl(jeopardy_cat_rows, num_question_jeopardy)  # pd.Series Type
    # DataFrame storing the randomly selected rows (question) for each randomly selected categories
    questions_jeopardy = questions_jeopardy.append(jeopardy_rows, ignore_index=True)
# Adding the Not-answered column and set to true
questions_jeopardy['not_answered'] = True
# Sorting the Double Jeopardy round DataFrame by Category and value columns
questions_jeopardy = questions_jeopardy.sort_values(by=['Category', 'Value']).reset_index(drop=True)
```

Code lines: 694 jeopardy.py

```
round_mode(jeopardy_cat, jeopardy_cat_list, questions_jeopardy, 'Jeopardy!', score, cheat_mode)
```

Section-2:
jeopardy.py - Round Mode

2.1 overview

The Round Mode is the code flow from start to game over .

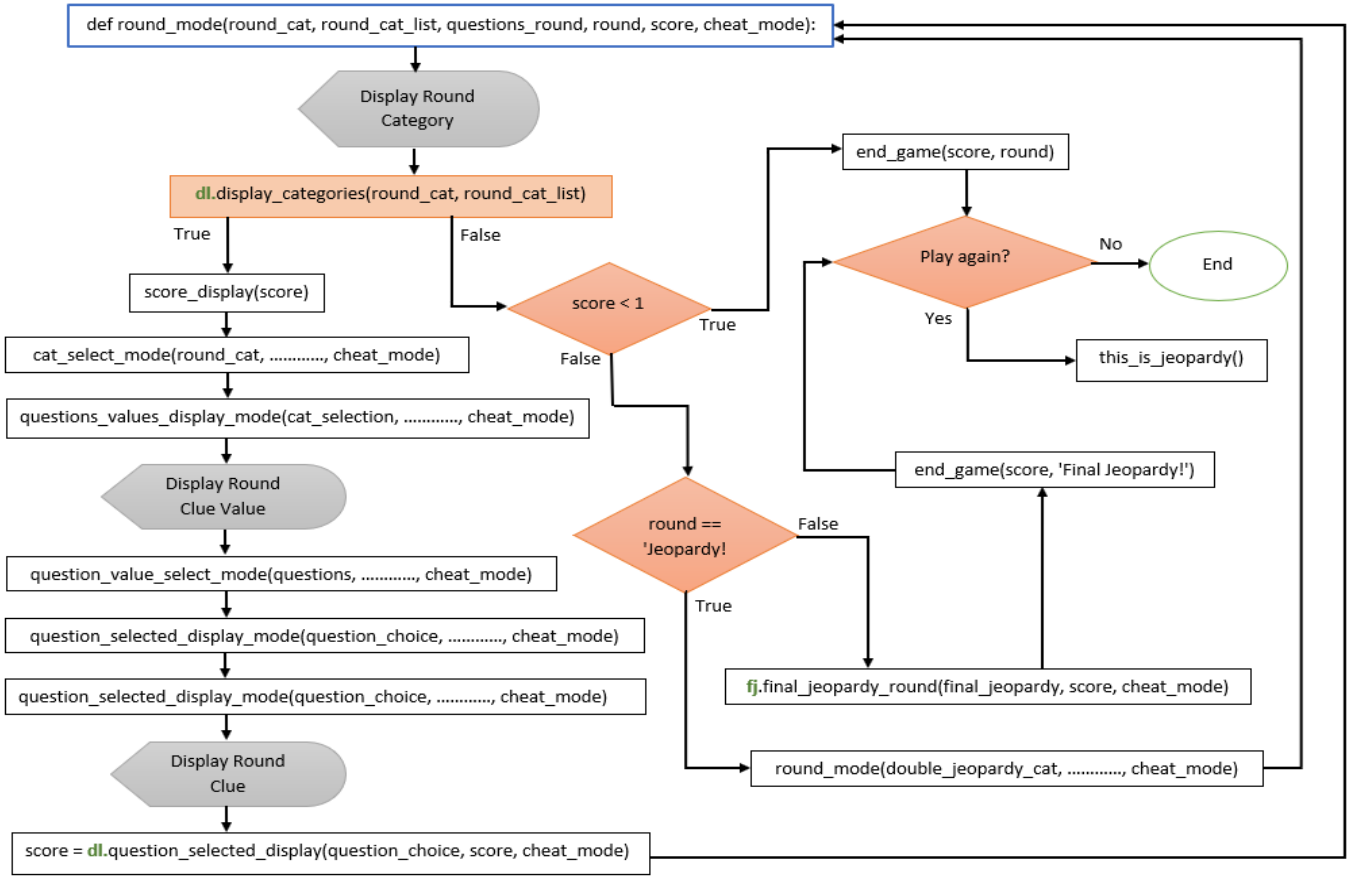
Jeopardy! and Double Jeopardy! Rounds

- Display and Select categories
- Display the questions values
- Select a question value
- Display the selected question and check the user answer
- Check if all the questions in all the categories have been answered and redirects to next the round
- Launch the Final Jeopardy! round from final_jeopardy.py
- Quit the game

All the Rounds

- Display the score
- End the game

2.2 Round Mode Flowchart



2.3.0 The round_mode() function

The round_mode() function is both a Jeopardy! and Double Jeopardy! Rounds function, it displays the category banner and the randomly selected categories.

The function dl.display_categories(round_cat, round_cat_list) returns False or True.
If True is returned, the function displayed the selected categories, and the code proceeds to display the user score and calls the category selection mode function, cat_select_mode(), else see next slide.
The function dl.display_categories() is part of the game code sub-section display_lib.py.

Code lines: 509-529 jeopardy.py

```
def round_mode(round_cat, round_cat_list, questions_round, round, score, cheat_mode):
    # Wipe console display
    dl.wipe_screen()
    # Category display banner
    print('\n*****************************************')
    print('  You are in the ' + round + ' round')
    print('*****************************************')
    print('''
********************************************************************************
**                                                                            **
**                        ----- CHOOSE A CATEGORY -----                       **
**                                                                            **
********************************************************************************''')
    print('\n')
    # Displays the categories,
    # and return True if all the questions in all the categories haven been not-answered
    if dl.display_categories(round_cat, round_cat_list):
        score_display(score)
        # Redirects to the user category selection function
        cat_select_mode(round_cat, round_cat_list, questions_round, round, score, cheat_mode)
……….
```

2.3.1 The round_mode() function

If the function dl.display_categories() returns False, meaning that for that particular round, all the questions in all the categories have been answered, the player, depending of her/his score, is redirected to the next round or to the end of game, the game is over.

Code lines: 530-551 jeopardy.py

```
……..
    #
    print('''
**********************************************************************

All the clues in the all categories have been answered.

**********************************************************************

Press any key to continue

**********************************************************************''')
    msvcrt.getch()
    # Checks score if to proceed to the next round
    if score < 1:
        end_game(score, round)
    # Next double Jeopardy Round
    elif round == 'Jeopardy!':
        round_mode(double_jeopardy_cat, double_jeopardy_cat_list, questions_double_jeopardy, 'Double Jeopardy!', score, cheat_mode)
    else:
        # Next Final Jeopardy Round
        fj.final_jeopardy_round(final_jeopardy, score, cheat_mode)
        # Game ends
```

2.4 The cat_select_mode() function

The cat_select_mode() function is a Jeopardy! and Double Jeopardy! Rounds function, it manages the user category choice input.

The user has the choice to quit the game by pressing “q”. If “q” is pressed, the user will be redirected to the quit game menu, dl.quit_game(). If not she/he will be redirected to the category question values display, questions_values_display_mode().

The function ie.check_error_cat() checks for user error inputs, the function is part of the game code sub-section input_error.py.
The function dl.quit_game() is part of the game code sub-section display_lib.py.

Code lines: 47-501 jeopardy.py

```
  def cat_select_mode(round_cat, round_cat_list, questions_round,
                                round, score, cheat_mode):
        # User selects Category
        print()
        print('''
************************
 Choose a category!
 Or press Q to quit.
************************''')
        # User input
        category_choice = msvcrt.getch()  # getch returns a byte data type
        # b'q', the b stands for byte
        if category_choice.lower() == b'q':
            # Quite?
            dl.quit_game()
            # user replied no to leave the game
            print('\n************************\n\n Please choose a category')
            category_choice = msvcrt.getch()
        # Checks for user invalid key inputs
        category_choice = ie.check_error_cat(round_cat, category_choice, round_cat_list)
        # Returns the selected category
        cat_selection = (round_cat.Category[category_choice - 1])
        # Redirects to the question values menu for selected category
        questions_values_display_mode(cat_selection , round_cat, round_cat_list,
                                              questions_round, round, score, cheat_mode)
```

2.5 The questions_values_display_mode() function

The questions_values_display_mode() function is both a Jeopardy! and Double Jeopardy! Rounds function, it displays the question values banner and the selected category question values.

After displaying the banner and the values, the function calls the question_value_select_mode() function.

The function dl.question_values_display() displays the category question values, the function is part of the game code sub-section display_lib.py.

Code lines: 446-472 jeopardy.py

```
def questions_values_display_mode(cat_selection, round_cat, round_cat_list,
                questions_round, round, score, cheat_mode):
    # Wipe console display
    dl.wipe_screen()
    # Questions values display banner
    print('\n*****************************************')
    print('You are in the ' + round + ' round')
    print('*****************************************')
    if round == 'Jeopardy!':
        print('''
**********************************************************************
**                                                                  **
**                      ----- CHOOSE A CLUE VALUE -----             **
**                                                                  **
**********************************************************************''')
    else:
        print('''
**********************************************************************
**                                                                  **
**                      ----- CHOOSE A CLUE VALUE -----             **
**  The clue values are Doubled                                     **
**********************************************************************''')
    print()
    # Displays the questions for the selected category checks
    # if the question was already answered
    # and returns the questions DataFrame of the selected category
    questions = dl.question_values_display(questions_round, cat_selection, score)
    # Redirects to the user questions selection mode
    question_value_select_mode(questions, round_cat, round_cat_list,
                questions_round, round, score, cheat_mode)
```

2.6 The question_value_select_mode() function

The question_value_select_mode() function is a both a Jeopardy! and Double Jeopardy! Rounds function, it manages the user question value choice input.

The use has the choice to quit the game by pressing “q”. If “q” is pressed, the user will be redirected to the quit game menu, dl.quit_game(). If not she/he will be redirected to the selected question display mode, question_selected_display_mode().

The function ie.check_error_question() checks for user error inputs, the function is part of the game code sub-section input_error.py.
The function dl.quit_game(), is part of the game code sub-section display_lib.py.

Code lines: 408-439 jeopardy.py

```
 def question_value_select_mode(questions, round_cat, round_cat_list,
                questions_round, round, score, cheat_mode):
        # User selects question
        print()
        print('''
************************
Choose a clue value!
Or press Q to quit.
************************''')
        # User input
        value_choice = msvcrt.getch()  # getch returns a byte data type
        # b'q', the b stands for byte
        if value_choice.lower() == b'q':
            # Quite?
            dl.quit_game()
            # user replied no to leave the game
            print('\n************************\n\n Please choose a clues value')
            value_choice = msvcrt.getch()
            # Checks for user invalid key inputs
        value_choice = ie.check_error_question(value_choice, questions)
        # Sets question as answered
        questions.loc[[value_choice - 1], 'not_answered'] = False
        questions_round.loc[questions_round['Question'] == questions.Question[value_choice - 1], \
                            'not_answered'] = False
        # Checks if all the questions in the category have been answered
        if any(questions['not_answered']) == False:
            # Sets all question in category to answered
            round_cat.loc[round_cat['Category'] == questions.Category[value_choice - 1], \
                          'questions_not_answered'] = False
        # Returns the row of the selected question
        question_choice = questions.loc[[value_choice - 1]]
        # Redirects to the question display mode
        question_selected_display_mode(question_choice, round_cat, round_cat_list,
                                       questions_round, round, score, cheat_mode)
```

2.7 The question_selected_display_mode() function

The question_selected_display_mode() function is both a Jeopardy! and Double Jeopardy! Rounds function, it displays the question banner and the user selected question.

After displaying the banner and the values, the function recalls the round_mode() function.

The function dl.question_selected_display() displays the question, handles the user answer input and returns the player new score. The function is part of the game code sub-section display_lib.py.

Code lines: 383-401 jeopardy.py

```
def question_selected_display_mode(question_choice, round_cat, round_cat_list,
                questions_round, round, score, cheat_mode):
    # Wipe console display
    dl.wipe_screen()
    # Question selected banner
    print('\n*****************************************')
    print('  You are in the ' + round + ' round')
    print('*****************************************')
    print('''
********************************************************************************
**                                                                            **
**                            ----- CLUE -----                                **
**                                                                            **
********************************************************************************''')
    print()
    # Displays the selected question, checks the user answer
    # and returns the new score
    score = dl.question_selected_display(question_choice, score, cheat_mode)
    # Redirect to round mode
    round_mode(round_cat, round_cat_list, questions_round, round, score, cheat_mode)
```

2.8 The end_game() function

The user reached the end of the game.

The end_game() function displays the user score and the Round reached.

The user has the choice to replay.

Code lines: 42-67 jeopardy.py

```
def end_game(score, round):
    if round != 'Final Jeopardy!':
        print('-------------------------------------------------------------------------------\n')
        print('     Your winnings are $' + str(score) + '\n')
        print('     You have no winnings, you can not proceed to the next round... :(\n \n  \
        You reached the Round: ' + round)
    print('''
--------------------------------------------------------------------------

Do you want to play again?

--------------------------------------------------------------------------

Press Y for yes and N for No

--------------------------------------------------------------------------
                    ''')
    key = msvcrt.getch()
    if key.lower() == b'y':
        this_is_jeopardy()
    elif key.lower() == b'n':
        exit()
    else:
        print('---------------------- error\n wrong key.........')
        # Calls back
        end_game(score, round)
```

Section-3: display_lib.py

3.1 overview

The display_lib.py is mostly the game console output functions library for both the Jeopardy and Double Jeopardy Rounds, the Final Jeopardy! Round only utilizes the display score and wipe the console screen clear functionalities from the library.

It is a game code sub-section utilized by jeopardy.py and final_jeopardy.py

- Display categories
- Display the questions values
- Display the selected question and check the user answer
- Quit the game
- Display the score
- Wipe the console screen clean

3.2 Importing Libraries

Code lines: 22-26 display_lib.py

```
import re # regex
# Miscellaneous operating system interfaces
import os # I used os.system('cls') to wipe the console display
# getch() for windows operating system
import msvcrt
```

os provides a portable way of using operating system dependent functionality. If you just want to read or write a file use open(), for example.
For more information about os:
https://docs.python.org/3/library/os.html

3.3 The display_categories() function

The function display_categories(), will, if all the questions in all the categories have been answered, any(df_cat['questions_not_answered']) == False, returns False, else it will return True and displays categories having at least one question not answered, df_cat['questions_not_answered'][i - 1] == True.

Code lines: 152-168 display_lib.py

```
# Displays the categories for both the Jeopardy and Double Jeopardy rounds
# and checks all questions if answered
def display_categories(df_cat, df_list):
    if any(df_cat['questions_not_answered']):
        i = 0  # Tracks choice numbers
        # displaying Categories
        print(
            '**********************************************************************************************************\n')
        for category in df_list:
            i += 1
            # Checks if all the questions in a category haven been not-answered
            if df_cat['questions_not_answered'][i - 1]:
                print('                              Type -' + str(i) + '- for category: ' + category)
        print(
            '\n**********************************************************************************************************')
        # Returns True if any of the questions in the category are not answered
        return True
    # Returns False if all of the questions in the category are answered
    return False
```

3.4 The question_values_display() function

The function question_values_display(), creates a DataFrame, questions, containing the questions rows from the user selected category, displays the not-answered question values, questions['not_answered'][i - 1] == True, and returns the questions DataFrame.

Code lines: 123-144 display_lib.py

```
def question_values_display(questions_df, choice, score):
    # Creating a data Storing the rows for the selected category
    questions = questions_df.loc[questions_df.Category == choice]
    questions = questions.sort_values(by=['Category', 'Value']).reset_index(drop=True)
    # From a pd.Series type to an object list type to be used in a for loop
    questions_list = list(questions.Question)
    # Displays question and category
    print(
        '*********************************************************************************************************\n')
    print(' From Category: ' + choice)
    print()
    i = 0  # Choice number
    for q in questions_list:
        i += 1
        # Checks if the question has been not-answered
        if questions['not_answered'][i - 1]:
            print('                                Type -' + str(i) + '- : $' + str(int(questions.Value[i - 1])))
    print(
        '\n*********************************************************************************************************')
    score_display(score)
    # Returns the DataFrame questions rows for the selected category
    return questions
```

3.5.0 The question_selected_display() function, cheat_mode

The function question_selected_display() displays the selected question, checks the user answer and returns the player new score.

If cheat_mode == True, the function will display the question answer.

Code lines: 68-84 display_lib.py

```
# Displays the selected question, checks the user answer
# and returns the new score
def question_selected_display(question_row, score, cheat_mode):
    # question_row is 1 row DataFrame
    # We need to reset the question_row index, we want the row index = 0
    question_row = question_row.reset_index(drop=True)
    # Displays question and category
    print('************************************************************')
    print('\n From Category: ' + str(list(question_row.Category)[0]))
    print('\n For a value of: $' + str(int(list(question_row.Value)[0])))
    print('\n Clue:\n ' + str(list(question_row.Question)[0]))
    print()
    score_display(score)
    print()
    if cheat_mode:
        print('************************************************************')
        print(' You are on cheat mode...\n The right response is:')
        print(' ' + str(list(question_row.Answer)[0]))
    print('************************************************************\n')
```

……..

3.5.1 The question_selected_display() function, answer

After the function displays the question, the user can input her/his answer, user_answer = input().

The regex function re.search() compares the user answer with the question answer.
If re.search(answer, user_answer) = True, the question value is added to the player winnings, else the question value is subtracted from the player winnings.

The function question_selected_display() returns the new player score.

Code lines: 88-115 display_lib.py

```
    print('''
**********************
 Type your response
**********************''')
    user_answer = input()
    # Compares the user answer with the right answer
    user_answer = user_answer.lower()
    answer = list(question_row.Answer)[0].lower()
    if re.search(answer, user_answer):
        score += question_row.Value[0]
        print('\n*****************************************************************')
        print('\n Right response :)\n Your winnings are now: $' + str(score))
        print('\n*****************************************************************\n')
        print('''
******************************
  Press a key to continue
******************************''')
        msvcrt.getch()
    else:
        score -= question_row.Value[0]
        print('\n*****************************************************************')
        print('\n Wrong response :(\n Your winnings are now: $' + str(score))
        print('\n The right response was: ' + question_row.Answer[0])
        print('\n*****************************************************************\n')
        print('''
******************************
  Press a key to continue
******************************''')
        msvcrt.getch()
        # Returns the new score
    return score
```

3.6 The quit_game() and score_display( ) functions

The quite_game() function allows the user to quit the game, exit().
Note: that msvcrt.getch() from the library msvcrt is used to capture the user character input, the b in b'y' stand for byte base data.

The score_display() function displays the user score, when called

Code lines: 46-60 display_lib.py

```
def quit_game():
    print()
    print('''
**********************************************
Are you sure that you want to quite Jeopardy?
Press Y for yes and N for No
**********************************************''')
    q_choice = msvcrt.getch()
    if q_choice.lower() == b'y':
        exit()
    elif q_choice.lower() == b'n':
        return
    else:
        print(' Please Press Y for Yes and N for No!')
        quit_game()
```

Code lines: 42-56 display_lib.py

```
def score_display(score):
    print()
    print(' Your winnings are $' + str(score))
```

3.7 The wipe_screen() function

The wipe_screen() function wipes the console screen clear.

The function utilizes the os library function os.system() with an if statement to check the type of operating system used by the computer and clears the console screen.

Code lines: 34-36 display_lib.py

```
def wipe_screen():
    # Checks if the operating system is MS, and clear the console display
    os.system('cls' if os.name == 'nt' else 'clear')
```

Section-4: final_jeopardy.py

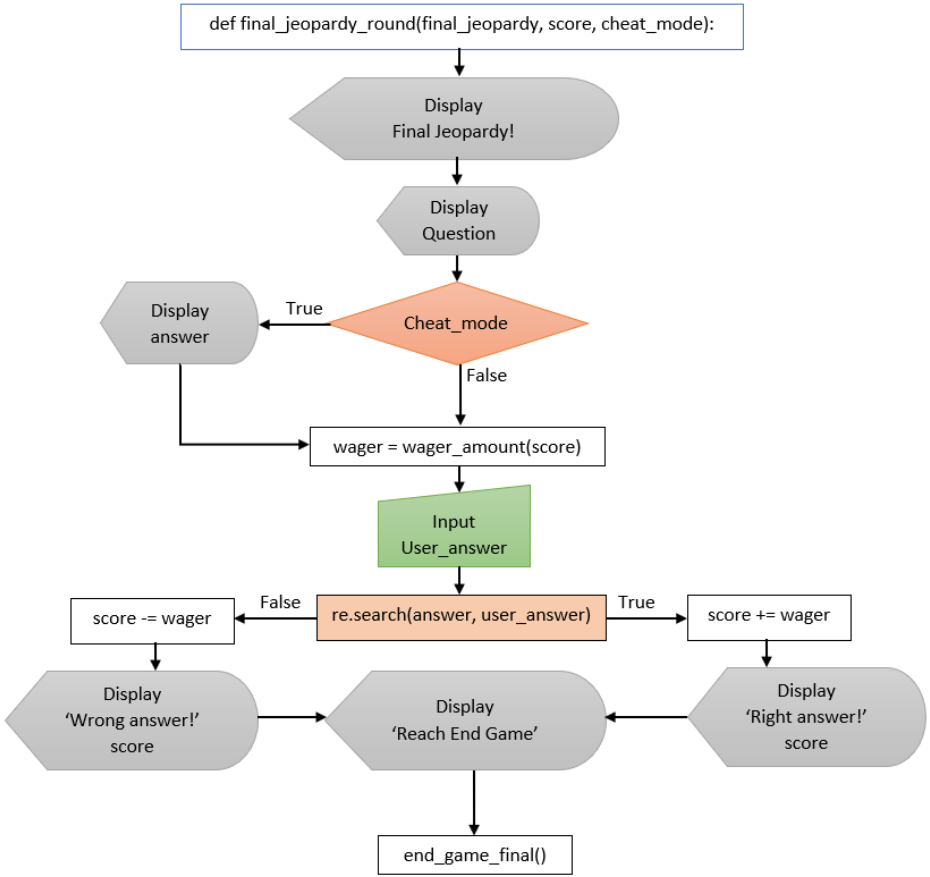
4.1 overview

The final_jeopardy.py is the code section running the Final Jeopardy! Round.

It is a game code sub-section utilized by jeopardy.py

- Display question
- Take a wager
- User input answer
- Compare the user answer with the question answer
- Display the score
- End game

4.2 Final Jeopardy! Round Flowchart



4.3 Importing Libraries

Code lines: 19-21 final_jeopardy.py

```
#
import re # regex
import msvcrt # getch()
```

re, Regular expression operations, provides regular expression matching operations similar to those found in Perl.
For more information about re:
https://docs.python.org/3/library/re.html

4.4 Importing Game Files

The fowling functions , from the other game code sub-sections, are utilized by final_jeopardy.py.

input_error.py **function:**
- check_int_error()

display_lib.py **function:**
- wipe_screen()

Code lines: 27-30 final_jeopardy.py

```
# Input errors handling functions
import input_error as ie
# Display function file
import display_lib as dl
```

4.5.0 The final_jeopardy_round() function, wager

After the function displays the question, the user can input a wager, wager = wager_amount(score).

Code lines: 66-97 final_jeopardy.py

```
def final_jeopardy_round(final_jeopardy, score, cheat_mode):
    # Wipe console display
    dl.wipe_screen()
    # Final Jeopardy Banner
    print()
    print('''
**********************************************************************************
**                                                                              **
**                          Congratulations you reached                         **
**                            Final Jeopardy Round-3                            **
**                                                                              **
**********************************************************************************
                ''')
    # Displays question and category
    print('\n************************************************************')
    print('\n From Category: ' + str(list(final_jeopardy.Category)[0]))
    print('\n Clue:')
    print(' ' + str(list(final_jeopardy.Question)[0]))
    print()
    dl.score_display(score)
    print('\n************************************************************\n')
    if cheat_mode:
        print('\n************************************************************')
        print(' You are on cheat mode...\n The right response is:')
        print(' ' + str(list(final_jeopardy.Answer)[0]))
    print('************************************************************\n')
    print('''
******************
Enter your wager
******************''')
    # Returns the user input wager amount and check for errors
    wager = wager_amount(score)
```

4.5.1 The final_jeopardy_round() function, answer

After the user inputs her/his answer, user_answer = input(' '), the function compares the user answer with the question answer.

If re.search(answer, user_answer) == True, the user wager is added the user score, else it is subtracted from the user score.

Code lines: 98-126 final_jeopardy.py

```
**********************
Type your response
**********************''')
    user_answer = input(' ')
    # Compares the user answer with the right answer
    user_answer = user_answer.lower()
    answer = list(final_jeopardy.Answer)[0].lower()
    if re.search(answer, user_answer):
        score += wager
        print('\n*****************************************************************\n')
        print(' Right response :)\n Your winnings are now: $' + str(score))
        print('\n*****************************************************')
        print('''
******************************
Press any key to continue
******************************''')
        msvcrt.getch()
    else:
        score -= wager
        print('\n*****************************************************************\n')
        print(' Wrong response :(\n Your winnings are now: $' + str(score))
        print('\n The right answer was: ' + list(final_jeopardy.Answer)[0])
        print('\n*****************************************************')
        print('''
******************************
Press any key to continue
******************************''')
        msvcrt.getch()
```

4.5.2 The final_jeopardy_round() function, end game.

Code lines: 129-141 final_jeopardy.py

After displaying the congratulation banner and the user score the function ends and the code redirects the user to the Jeopardy.py end_game(score, 'Final Jeopardy!') function, the game ends.

```
dl.wipe_screen()
print('''
********************************************************************************
**                                                                            **
**                    Congratulations you reached                             **
**                       The end of the Game                                  **
**                                                                            **
********************************************************************************''')
print('\n********************************************************************************\n')
print('                     Your Total winnings are : \n\n                          $' + str(score) + '\n')
print('\n********************************************************************************\n')
# End Game
# Jeopardy.py end_game(score, 'Final Jeopardy!')
```

4.6 The wager_amount() function

The wager_amount() function take the user wager input and checks for input errors.

After the user enters a wager,
wager = input(' '), the function checks if the input is a character or an integer, and if the wager amount is superior than the player winnings.

The function ie. check_num_error() checks if the user input is an integer, the function is part of the game code sub-section input_error.py.

Code lines: 98-126 final_jeopardy.py

```
def wager_amount(score):
    wager = input(' ')
    # Checks if the User is entering a no-integer
    while ie.check_num_error(wager):
        print('\n-------------------------------- Error')
        print(' Wrong entry')
        print('-------------------------------- Error')
        print('\n************************\n Enter a whole number amount\n')
        #
        wager = input(' ')
    # Checks if an invalid numeric key was pressed by user
    if int(wager) > score:
        print('\n-------------------------------- Error')
        print(' Wrong entry')
        print('-------------------------------- Error')
        print('\n************************\n The amount is greater than yor winning')
        print(' Please enter an amount same as or lesser than your winning \n')
        # calls back
        wager = wager_amount(score)
    # Returns
    return int(wager)
```

Section-5: inpute_error.py

5.1 overview

The input_error.py is the code section handling user error inputs Round.

It is a game code sub-section utilized by jeopardy.py and final_jeopardy.py

- Check for user choice category input errors
- Check for user choice question value input errors
- Category and Question value user choice error messages
- getch() input exception ValueError
- input() input exception ValueError

5.2 Importing Libraries

Code lines: 21 input_error.py

```
# getch() for windows operating system
import msvcrt
#
```

Again
msvcrt provides access to some useful capabilities on Windows platforms, in this project the following functionality for Console I/O were used:

- msvcrt.getch()
  Read a keypress and return the resulting character as a byte string. Nothing is echoed to the console. This call will block if a keypress is not already available but will not wait for Enter to be pressed.

For more information about msvcrt:
https://docs.python.org/3/library/msvcrt.html

If you are interested to make the Console I/O character capturing compatible on multiple platforms:
https://docs.python.org/3/library/curses.html#curses.window.getch
https://docs.python.org/3/howto/curses.html#curses-howto

Note:
If you are using the IDE Pycharm's console, msvcrt.getch() will not work, you need to use the window command prompt console.

5.3 The check_error_cat() function

The function checks for user category choice input errors.

It checks for user input ValueErrors,
check_num_error(category_choice),

for user wrong number input,
category_choice == 0 or category_choice > len(round_cat_list),

and if the category number pressed by the user is a number associated with a category having all its questions already answered,
category_df['questions_not_answered'][category_choice - 1] == False

Code lines: 101-118 input_error.py

```
def check_error_cat(category_df, category_choice, round_cat_list):
    # Checks if a no-numeric key was pressed by user
    while check_num_error(category_choice):
        category_error_message()
        category_choice = msvcrt.getch()

    category_choice = int(category_choice)
    # Checks if an invalid numeric key was pressed by user
    if category_choice == 0 or category_choice > len(round_cat_list):
        category_error_message()
        category_choice = msvcrt.getch()
        # Calls Back
        category_choice = check_error_cat(category_df,
                                category_choice, round_cat_list)
        # Checks if the select category questions have been already answered
    if category_df['questions_not_answered'][category_choice - 1] == False:
        print('------------------------------------------------- Error')
        print('   All the clues in ')
        print('   The category -' + str(category_choice) + '- ' + \
              category_df.Category[category_choice - 1])
        print('   have been answered...')
        print('------------------------------------------------- Error')
        print('   Please choose another category')
        print('-------------------------------------------------')
        category_choice = msvcrt.getch()  # getch returns a byte data type
        # Calls back
        category_choice = check_error_cat(category_df,
                                category_choice, round_cat_list)
    # Returns category_choice as integer 0<>4
    return category_choice
```

5.4 The check_error_question() function

The function checks for user question choice input errors.

It checks for user input ValueErrors,
check_num_error(question_choice),

for user wrong number input,
question_choice == 0 or question_choice > len(questions),

and if the question number pressed by the user is a number associated with a question already answered,
questions['not_answered'][question_choice - 1] == False

Code lines: 69-95 input_error.py

```
def check_error_question(question_choice, questions):
    # Checks if a no-numeric key was pressed by user
    while check_num_error(question_choice):
        question_error_message()
        question_choice = msvcrt.getch()
    # From byte type to integer type
    question_choice = int(question_choice)
    # Checks if an invalid numeric key was pressed by user
    if question_choice == 0 or question_choice > len(questions):
        category_error_message()
        question_choice = msvcrt.getch()
        # Calls Back
        question_choice = check_error_question(question_choice, questions)
        # Checks if the select category questions have been already answered
    if questions['not_answered'][question_choice - 1] == False:
        print('-------------------------------------------------- Error')
        print('   The clue value -' + str(question_choice) + '- ' + \
              questions.Question[question_choice - 1])
        print('    has been already answered...')
        print('-------------------------------------------------- Error')
        print('   Please choose another clue value')
        print('--------------------------------------------------')
        question_choice = msvcrt.getch()  # getch returns a byte data type
        # Call back
        question_choice = check_error_question(question_choice, questions)
    # Returns question_choice as integer 0<>4
    return question_choice
```

5.5 Category and question message errors

The functions, question_error_message() and category_error_message(), display an error message and they are respectively associated with the functions check_error_question() and check_error_category().

Code lines: 57-63 input_error.py

```
#
# ---- Question selection error message
#
def question_error_message():
    print('\n------------------------------- Error')
    print(' Wrong entry\n')
    print('------------------------------- Error')
    print('\n************************\n Please choose a clues')
#
# ---- Category selection error message
#
def category_error_message():
    print('\n------------------------------- Error')
    print(' Wrong entry\n')
    print('------------------------------- Error')
    print('\n************************\n Please choose a category')
```

5.6 Exceptions error handling function

The function, checks for user input ValueErrors.
It returns True if the input is a numeric key input with getch() or an integer with input(), else it returns False.

Code lines: 30-36 input_error.py

```
def check_num_error(key):
    try:
        key = int(key)
    except ValueError:
        return True
    # User entered a no-integer input
    return False
```

Conclusion

The Jeopardy Console Game is a personnel project, an offshoot from the Codecademy Data Science path, under the Data Analysis with Pandas Phyton library section.

It is a Python Pandas library based, solo player Jeopardy console game.

Features:

- Tidying data from the jeopardy.csv file to be used by the console game.
- Data manipulation with the python Pandas library.
- User input with msvcrt.getch().
- Error handling.

Implementation:

- Description option
- Cheat mode
- Settings option
- Round Mode for the Jeopardy! and Double Jeopardy Rounds
- Final Jeopardy Round

The project can be improved by adding a player name/score record feature and a highest score bord display feature.

Project Overview

The Jeopardy Console Game is a personnel project, an offshoot from the Codecademy Data Science path, under the Data Analysis with Pandas Phyton library section.

It is a Python Pandas library based, solo player Jeopardy console game.

The project features:

- Tidying data from a csv file to be used by the console game.
- Data manipulation with the python Pandas library.
- User input with msvcrt.getch().
- Error handling.