CSC 122 Python Applications

Lesson 3: Project 1-B – Wake Golf Tour

Objectives

In this assignment, students will:

- Use basic Python object-orientated programming constructs, including creating classes, creating objects and calling their methods.
- Code Python functions that read list data and place the data into class objects.
- Learn how to carefully read a program design specification and translate it into code.

Python Programming Instructions

Please use PyCharm or a text editor (notepad++ or IDLE are good for Python) to type your program code files. You can use PyCharm or the command line to test your program. See documents in Course Resources for more details about these environments.

a. If you are using **PyCharm**, then name the project exactly **WakeGolfTourB**. The Python source code will be in the **'WakeGolfTourB'** folder. Place the Python program file, **golf_tour.py**, the six class definitions files (**golfCourse.py**, **hole.py**, **golfer.py, tournament.py, round.py,** and **tournGolfer.py**) and the **input files** in the **WakeGolfTourB** project folder. The output files should be in the folder after running the program.

b. If you are using the **command line interface**, create a **WakeGolfTourB** folder. Place the Python program file, **golf_tour.py**, the six class definitions files (**golfCourse.py**, **hole.py**, **golfer.py, tournament.py, round.py,** and **tournGolfer.py**) and the **input files** in the **WakeGolfTourB** project folder. Run the program from that folder. The output files should be in the folder after running the program.

Once you have completed the assignment, zip up the **WakeGolfTourB** folder using directions from the document, **Creating and Submitting Programs**. Submit the **WakeGolfTourB.zip** file to Blackboard for credit.

Programs that are submitted incorrectly will not be graded.

Program Specifications

Project Description

Please re-read the **Wake Golf Tour App** and **Wake Golf Tour Specification** documents. Project 1 has four-parts that follow each other. The code for Project 1-B starts with the code from Project 1-A, and the code for Project 1-C starts with the code from Project 1-B, and so on.

You are required to follow the specifications exactly as given. Algorithms will be provided for each of the functions. In addition, the code for some of the functions will be provided. **You must follow the algorithm and code for the functions for which the code is given to gain an understanding of how the related functions are coded.** Then code up the functions for which you must provide your own code, using the given algorithms.

Project Part B:

This second part of the project, **Project 1-B**, builds on the Python program file called **golf_tour.py** from **Project 1-A**. It also adds three *new* class definition files, **tournament.py, round.py,** and **tournGolfer.py**. The main program contains 8 functions, including the **main** function. Details about these functions are given to you in comments within the functions themselves and are summarized in the **Wake Golf Tour Specification** document.

In Project 1B, you are to create the **Tournament**, the **TournGolfer** and the **Round class definitions** and **object lists**. You are to read the **Tournament** data from the **input CSV file**, which is used to create the **Tournament objects list**. The **create_tournaments** function should also create and return a dictionary, **tourn_golfers_dict,** to be used as input data for creating the **TournGolfer objects list**. Each of the **create functions** produce a list of class objects containing the data for a specific database table, including the **id** fields. This data is displayed on the screen and returned to be used as an input parameter to other **create functions** and to be saved in a file. The object data is written to the screen and a file using the class string (**__str__**) method.

New Create Functions

create_tournaments (input_filename, golf_courses_list):
tourns_list, tourn_golfers_dict
create_rounds (tourns_list): rounds_list

create_tourn_golfers (tourn_golfers_dict, golfers_list):
tourn_golfers_list

Provided Code

The **golf_tour.py** program has the following function and class definition code provided for you from Project1-A. **Do not change the code in these functions**:

- **main**
- **create_golf_courses**
- **write_objs_to_file**
- **GolfCourse** in **golfCourse.py**

Required Code

You must copy your existing code from Project 1-A for:

create_golfers	- From Project 1-A
create_holes	- From Project 1-A
Hole in **hole.py**	- From Project 1-A
Golfer in **golfer.py**	- From Project 1-A

Helpful Hint: After you copy your existing code from Project 1-A, run **golf_tour.py** to be sure it runs without errors (exit code 0) and generates the correct output. This will assure that you have working code before you start writing the new code required for Project 1-B.

You are responsible for coding up the following **new** functions. See the Appendix for example algorithms for these functions.

- **create_tournaments**
- **create_rounds**
- **create_tourn_golfers**

You will also need to complete the class functions in these **new** class files:

- **Tournament** in **tournament.py**
- **TournGolfer** in **tournGolfer.py**
- **Round** in **round.py**

Program Starter Code

There is a zipped folder in Blackboard called **'Project1BStarterCode.zip'** that contains the **input files**, the starter **golf_tour.py** program and the class definition files, **tournament.py, round.py,** and **tournGolfer.py**, for **Project 1-B**. The three input files are in comma-delimiter format (*.CSV), where each record has its fields separated by commas. Each of the provided files should be placed in the same directory as your program, **golf_tour.py**.

Program Input Files

golf_courses_infile = "golfCoursesInput.csv"
tournaments_infile = "tournamentsInput.csv"
golfers_infile = "golfersInput.csv"

Program Output Data

The new input data are read into the **golf_tour.py** program in the **create_tournaments** function and the data returned is used to create a list of objects containing that data. It also returns a dictionary, **tourn_golfers_dict,** which is used as input to the **create_tourn_golfers** function that uses it to create a list of objects containing the **tourn_ids** and **golfer_ids** linking the golfers to the tournaments in which they played.

Program Output Files

golf_courses_file = "golfCourses.csv"
holes_file = "holes.csv"
golfers_file = "golfers.csv"
tournaments_file = "tournaments.csv"
rounds_file = "rounds.csv"
tourn_golfers_file = "tournGolfers.csv"

Project Part B: create_tournaments function

In **create_tournaments,** step 1 is to create a lookup table dictionary for mapping the **golf_course_name** to the **golf_course_id** using the passed in **golf_course_list**. If you recall from Project1A, the **golf_course_list**, which was returned, is a list of the five golf course **objects**. We need to extract data (**golf_course_name** and **golf_course_id**) from these objects using the getters that were coded in the **GolfCourse** class. This extracted data is used to build the lookup table dictionary. This dictionary is used in this **create_tournaments** function to map the **golf_course_name** (given in the input file) to the **golf_course_id** (from the created dictionary).

Here is the code for creating the dictionary:

```
golf_course_name_to_id = dict()
for course in golf_course_list:
golf_course_name_to_id[course.get_course_name()] =
                          course.get_course_id()
```

Line 1: Create an empty dictionary: **golf_course_name_to_id**
Line 2: Traverse the GolfCourse objects in the **golf_course_list** – **course** holds a GolfCourse object
Line 3: The code fills in dictionary, where the key is **golf_course_name,** and the value is the **golf_course_id.** Both of these items are taken from the GolfCourse object using the getters.

This dictionary created with the code above is used later in the **create_tournaments** function to map the **golf_course_name** (given in the input file as the first piece of tournament information) to the **golf_course_id** (from the created dictionary) using the code below:

```
golf_course_id = golf_course_name_to_id[golf_course_name]
```

Use this same format in **create_tourn_golfers,** when creating a lookup table dictionary for mapping the **golfer_name** to the **golfer_id**.

Program Execution

Please re-read the **Wake Golf Tour App** and **Wake Golf Tour Specification** documents, until you have grasped the purpose of Project 1.

To Begin:

1. Please open the **Project1BStarterCode.zip** file, now!

2. Replace the **golfCourse.py**, **hole.py**, and **golfer.py** class definition files with the ones from Project 1-A

3. Copy the code for the **create_golfers** and **create_holes** from Project 1-A.

4. To better understand this project, organize document notes on paper, note cards, or on a whiteboard.

5. Come to Open Lab to get help.

6. Use the Students Helping Students Discussion Board in Blackboard to get help.

7. Email me questions and code to review, if you need help.

8. You may team up with others. Use the Students Helping Students Discussion Board in Blackboard to find partners.

You are not on your own. Your teacher, the lead instructor and other students can help you.

Program Output to Screen:

```
Wake Golf Tour Project 1

The GolfCourse object list:

< See Project 1-A output >

The Hole object list:

< See Project 1-A output >

The Golfer object list:

< See Project 1-A output >

The Tournament List object list:

1,Raleigh 1,1,2016-05-07,2,15
2,Raleigh 2,1,2016-06-09,4,15
3,Raleigh 3,1,2016-07-22,3,15
4,WTCC 1,2,2016-05-13,3,15
5,WTCC 2,2,2016-06-18,2,15
6,WTCC 3,2,2016-07-28,4,15
7,Garner 1,3,2016-05-19,4,15
8,Garner 2,3,2016-06-24,3,15
9,Garner 3,3,2016-08-06,2,15
10,Cary 1,4,2016-05-28,2,15
11,Cary 2,4,2016-07-08,3,15
12,Cary 3,4,2016-08-11,4,15
13,Apex 1,5,2016-06-03,3,15
14,Apex 2,5,2016-07-14,4,15
15,Apex 3,5,2016-08-20,2,15

The Round object list

1,1,Sat
2,1,Sun
```

```
3,2,Thu
4,2,Fri
5,2,Sat
6,2,Sun
7,3,Fri
8,3,Sat
9,3,Sun
10,4,Fri
11,4,Sat
12,4,Sun
13,5,Sat
14,5,Sun
15,6,Thu
16,6,Fri
17,6,Sat
18,6,Sun
19,7,Thu
20,7,Fri
21,7,Sat
22,7,Sun
23,8,Fri
24,8,Sat
25,8,Sun
26,9,Sat
27,9,Sun
28,10,Sat
29,10,Sun
30,11,Fri
31,11,Sat
32,11,Sun
33,12,Thu
34,12,Fri
35,12,Sat
36,12,Sun
37,13,Fri
38,13,Sat
39,13,Sun
40,14,Thu
41,14,Fri
42,14,Sat
43,14,Sun
44,15,Sat
45,15,Sun
```

```
The TournGolfer object list

1,1,1
2,1,2
3,1,3
4,1,4
5,1,5
6,1,6
7,1,7
8,1,8
9,1,9
10,1,10
...
...
220,15,10
221,15,17
222,15,24
223,15,1
224,15,8
225,15,15

Process finished with exit code 0
```

Appendix

Example algorithms

create_tournaments

```
"""
1. Create a lookup dictionary that contains the golf_course_name
   as the key and the golf_course_id as the value using the
   GolfCourse objects passed in the golf_course_list:
   See example code above.
2. a. Create an empty list called tournament_list
      that will be filled in with tournament objects
      created in this function from the input file data.
   b. Create an empty tourn_golfer_dict dictionary that
      will be filled in with the tournament id as the key
      and the list of golfers as the value.
      The loop below will fill in this dictionary value list,
      when each golfer name is read in.
3. Initialize the tourn_id to 1 and the tourn_id_key to 0
4. Use a try/except block to capture a File Not Found Error
   A. Open the input file object for reading the input file.
   B. Call the csv.reader function, passing in the input file
      and capturing the CSV file contents.
   C. Create a list from the file contents:
      tournament_input_list
   D. Create a loop to traverse the tournament_input_list,
      where the loop variable 'tourn_info' will contain either
      the tournament information, or a golfer name.
      Loop:
      1. Check the length of tourn_info; if length is greater
         than one, then process the tournament information
         a. Get each of the first five elements of the
            tourn_info list:
              Strip course_name, tourn_name, and start_date
              Convert num_rounds and num_golfers to ints.
         b. Get golf_course id from lookup dictionary created
            above
         c. Create a new Tournament object, call it tournament,
            passing in tourn_id, tourn_name,
            golf_course_id, start_date, num_rounds, and
            num_golfers
         d. Append the tournament object to the tournament_list
         e. Set the tourn_id_key to tourn_id
         f. Create dictionary entry value for this tourn_id_key,
            the value is an empty list to be filled in later
            with the golfer names as they are read from the
            input file.
         g. Increment the tour_id
      2. else the length of tourn_info is one, so this is a
         golfer name, add it to the tourn_golfers_dict value
         list. It will be used later in the create_tourn_golfers
         method.
         a. Get the golfer name from tourn_info stripping
            whitespace
         b. Add the golfer name to the tourn_golfers_dict value

   E. Close the input file
5. Print the tournament_list objects to the console
6. Return the tournament_list and tourn_golfers_dict
"""
```

create_rounds

```
"""
1. Create an empty list called rounds_list
   that will be filled in with Round objects
   whose data comes from the parameter - tournament_list
2. Initialize the round_id
3. Create an outer loop to traverse the input
   tournament_list, where the loop variable 'tourn'
   will contain one of the Tournament objects in
   tournament_list at each loop iteration
   Outer Loop
   a. Get the number_rounds and tourn_id from the
      Tournament object, tourn, and initialize
      num_rounds to number_rounds - this will be
      decremented below to find the correct day
      for the Round object being built
   b. Create an inner loop to run number_round times
      using the range function, where the loop
      variable 'r' keeps the count for the
      number of Rounds being created
      Inner Loop
      1. Check the value of num_rounds to determne
         the day value of this Round object.
         Use an if/elif/else structure to set the
         day instance variable
         if num_rounds == 4: day = "Thu"
            num_rounds == 3: day = "Fri"
            num_rounds == 2: day = "Sat"
            num_rounds == 1: day = "Sun"
      2. Decrement the num_rounds counter
      3. Create a Round object call it round passing
         in round_id, tourn_id, and day
      4. Append the Round object to the rounds_list
      5. Increment the round_id
4. Print the round objects to the console
5. Return the rounds_list
"""
```

create_tourn_golfers

```
"""
1. Create a lookup dictionary (golfer_name_to_id) for
   golfer_name to golfer_id
2. Create an empty list called tourn_golfers_list
   that will be filled in with TournGolfer objects
   whose data comes from the tournaments_list parameter,
   and object list parameter, golfers_list
3. Initialize the tourn_golfer_id
4. Create an outer loop to traverse the input
   tourn_golfers_dict, whose key will contain the
   tournament_id, and the value, 'golfer_name_list', will be the
   list of golfer names for that tournament
   Outer Loop
   a. Create an inner loop to traverse the
      golfer_name_list, where the loop variable 'golfer_name'
      will contain one of the golfer names for the tournament
      Inner loop
      1. get golfer_id from (golfer_name_to_id) lookup
         dictionary
      2. Create a TournGolfer object call it tourn_golfer,
         passing in tourn_golfer_id, tourn_id (from the dict
         key), and golfer_id
      3. Append the TournGolfer object to the
         tourn_golfers_list
      4. Increment the tourn_golfer_id

5. Print the tourn_golfers_list objects to the console
6. Return the tourn_golfers_list
"""
```