CIS 162 Project 3
Mega Millions Lottery Simulation

Due Date

- Phase 1 due Tuesday night, April 9 (11:59) in zybook (15 pts)
- Final submission Tuesday night, April 16 (11:59). Turn in printed document in lab.

Before Starting the Project

- Read chapters, 8, 9 and 10
- Read this entire project description before starting.

Learning Objectives

After completing this project you should be able to:

- *use* lists to process a collection of objects
- *use* loops
- *define* multiple classes
- *read data* from external text files

Mega Millions Lottery

Players pick five numbers (1-75) with no duplicates and a special number called the Mega Ball (1-15). Numbers are randomly generated twice per week to determine the winning amount on each ticket. Most tickets do not win anything! Almost no tickets earn more than $50. Here is the payout based on how many numbers match. But on rare occasions a lucky player gets $5 million!

# of matches	Mega Ball match	Prize
5	Yes	$5,000,000
5	No	$1,000,000
4	Yes	$5,000
4	No	$500
3	Yes	$50
3	No	$5
2	Yes	$5
1	Yes	$2
0	Yes	$1

Background Info: Reading Text Files in Python

Read section 10.1 for additional background. The data file contains fictional player and ticket information. A sample line of the file is shown below. The `readTickets()` method reads one line at a time and passes text to the `LotteryTicket` class constructor.

```
Amy, Zu, AnyCity, NC, 27834, 4/20/1960, 5, 12, 37, 39, 68, 11
```

The following sample code reads from a text data file one line at a time.

```
file = open(filename)
for entry in file:
    t = LotteryTicket(entry)
    self.add_ticket(t)
file.close()
```

Parsing Strings

Ticket information is contained in a long string with values separated by commas and slashes. You must write code that splits the string into individual pieces and converts them to integers and Strings as necessary. Blank spaces at the start and end of each String must be removed with the `strip()` method.

```
Amy, Zu, Phoenix, AZ, 78234, 4/20/1960, 4, 12, 15, 36, 67, 12
```

Here is sample code for LotteryTicket constructor you will need to adapt.

```
def __init__(self, info){

      # Split the info string into multiple pieces separated by ‘,’ or ‘/’
      tokens = info.split(",")
      self.first = tokens[0].strip()
      self.last = tokens[1].strip()

      # strip the full birthday and then extract the day, month, year
      birthday = tokens[5].strip()
      pieces = birthday.split(“/”)
      self.day = int(pieces[1])

      # resume with the six ticket numbers
      self.mega_ball = int(tokens[11].strip())
}
```

Step 1: Create a New Project in PyCharm

Step 2: Download Data File
Download the "ticket_info.txt" file from Blackboard and save it in the folder that PyCharm created for this new project. There are 50,000 entries!

Step 3: Create a new file called "lottery_ticket.py" (5 pts)
Within the file, write a class called `LotteryTicket`. Pay close attention to upper and lowercase for file names and class names. Maintain information about the ticket including: first name, last name, birth date (day, month, year), city, ST, zip code, a list of five numbers, mega ball (all ints) and a prize amount (float).

Constructor
A *constructor* is a special method that initializes instance members to appropriate starting values. Refer to section 9.6.

- `__init__(self, info)` - a constructor that initializes all of the instance variables to appropriate values given a single String. See information above about parsing Strings into smaller pieces. **Critical**, convert birth date values and lottery numbers to integers. Set initial prize to zero.
  ```
  Amy, Zu, Phoenix, AZ, 78234, 4/20/1960, 4, 12, 15, 36, 67, 12
  ```

Accessor Methods
Return the values of each instance member.
- `get_first(self)- return first name`
- `get_last(self)`
- `get_city(self)`
- `get_state(self)`
- `get_zipcode(self)`
- `get_day(self)`
- `get_month(self)`
- `get_year(self)`
- `get_prize(self)`
- `get_mega_ball(self)`
- `get_nums(self)` – return the list of five numbers.
- `has_ball(self, val)` – return `True` if `val` is one of the five numbers (but not the mega ball).
- `has_mega_ball(self, val)` – return `True` if `val` matches the mega ball.
- `__str__(self)` - return a formatted ticket summary (a single String with embedded newlines) . For example,
  ```
  Jane Smith
  San Francisco, CA 49401
  1 2 3 4 5   6
  Prize: $1.00
  ```

Mutator Methods

Modify the instance members.

- `set_prize(self, amount)` – set the prize amount

Test the class

```
if __name__ == '__main__':
    t = LotteryTicket("Louis,Laker,Allendale,MI,49401,4/20/1985,5,10,15,20,25,7")
    print(t)
```

Step 4: Create a new file called lottery_machine.py

Within the file, write a class called `LotteryMachine`. Pay close attention to upper and lowercase for file names and class names.

Import

Provide the following statement at the top of the file to use the LotteryTicket class.

```
from lottery_ticket import LotteryTicket
```

Instance Variables

- an instance variable that holds a list of `LotteryTicket`
- an instance member that holds a list of five numbers
- an instance member that holds the mega ball number

Constructor

A *constructor* is a special method that initializes instance members to appropriate starting values. Refer to section 9.6.

- `__init__(self)` – create an empty list to hold lottery tickets, an empty list to hold five lottery numbers, and an integer to hold the mega ball.

Accessor Methods

An *accessor* method does not modify class fields. The names for these methods, which simply return the current value of a field, often begin with the prefix 'get'.

- `get_ticket_count(self)` - return the number of tickets with one line of code.
- `get_mega_ball(self)` - return the current mega ball.
- `get_nums(self)` - return the list of five drawn numbers.

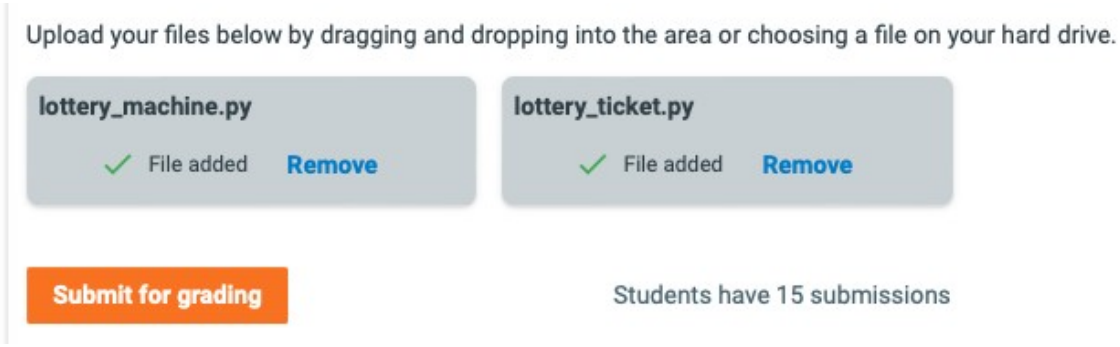
Initial Methods (10 pts)

A mutator method performs tasks that may modify instance members.

- `add_ticket (self, t)` – add the ticket to the `list` with one line of code.
- `read_tickets(self, filename)` - open the provided `filename` and read all the data. There are 50,000 entries in "ticket_info.txt". Refer to the information about reading text files earlier in this document.

Phase 1 Testing and Submission in zyBook

Perform a preliminary test by uploading "lottery_ticket.py" and "lottery_machine.py" to zyBook section 18.3. Note, you will not copy and paste code as you have in the past. The Submit for Grading button will not appear until both files are provided. Your solution should pass the first test if all functions have the correct name and perform correctly. **Note**, sometimes a file must be uploaded more than once for zyBook to recognize it.



Helper Methods (10 pts)
A helper method is used by other methods within the class but not called from outside the class.

- `draw_random_numbers(self)` – Select five random numbers between 1 - 75 without duplicates. Select the random mega ball (1-15) using `random.randint()`.

  Here is a primitive approach to select five numbers without duplicates. Create a temporary `list` of integers that holds numbers 1-75. Use `random.shulffle()` to mix the numbers and then repeatedly remove the first element of the list five times. Or, python provides a slick `random.sample()` function!
- `count_matches(self, t)` - return the number of matches between the selected five numbers (not the mega ball) and the provided ticket. Call `has_ball()` for the ticket object.
- `make_payouts(self)` – step through all tickets and assign the appropriate prize based on the selected six numbers. See award table. Invoke `count_matches()` to assist.
- `__str__(self)` - return a String that shows the six selected numbers.

```
Selected Numbers: 7 23 29 41 68   5
```

Mutator Methods (50 pts)
A mutator method performs tasks that may modify instance members.

- `draw_ticket(self)` – pick random numbers and assign awards to all tickets. This method only needs two lines of code because you will call `draw_random_numbers()` and `make_payouts()`.
- `test_ticket(self, b1, b2, b3, b4, b5, mega)` – this method is used for testing. Assign the parameters to the six lottery numbers and then call `make_payouts()`.
- `print_report(self, st)` – print a report for all tickets sold in the state with the two-letter abbreviation `st`. The report includes the six selected numbers, number of

tickets, average prize amount and the biggest winner. See below for sample output. Beware if no tickets were sold in the state. **Return** the number of tickets sold in the state.

- `get_oldest_player(self)` - return ticket of the oldest person in the database (to the day). Give careful thought to this one! Someone born Jan 15th is older than someone born Feb 3rd of the same year. If more than one person happens to share the title of oldest it does not matter which one you return.
- `get_big_winner(self)` - return the ticket with the largest prize. If more than one ticket happens to share the distinction then it does not matter which one you return.
- `get_big_winners(self, amount)` - return a list of all tickets with a prize of at least `amount`. If necessary, it is OK to return a list with no tickets. Hint, can be written in less than ten lines of code.
- `print_big_winners(self, amount)` – call `get_big_winners()` and then print all tickets. See below for sample output.

Additional Challenge (10 pts)

- `multiple_drawings(self, num)` – use a loop to repeatedly draw tickets `num` times or until a player wins the Mega Millions payout with six matches. After each drawing without a jackpot winner, increase the prize by $1,500,000. After all drawings are complete, **print** the number of drawings and biggest winner (see sample output). In case of a tie, it does not matter which player is reported. Consider adding instance variables for the jackpot amount and a boolean variable if a player wins the mega value. **Return** the largest prize value.

  Beware, python objects are assigned as references. Without going into too many details, you <u>should not</u> track the biggest winner as a `lottery_ticket` object. Instead, track the String representation of the object by calling the `__str__()` function.

```
winner = self.get_big_winner()
if bigger winner found:
   big_winner_name = winner.__str__()
   #update top prize value
```

Instructor Testing

Support instructor testing by including the following code at the bottom of your solution.

```
if __name__ == '__main__':
    lm = LotteryMachine()
    lm.read_tickets("ticket_info.txt")
    print(lm.multiple_drawings(100))
```

Final zyBook Testing
Perform final tests by uploading both files to zyBook. Your solution should pass all tests if all methods have the correct name and perform correctly. This does not necessarily mean everything works correctly. It is your responsibility to understand all requirements and perform your own testing to the extent possible.

Sample Results
Your output should look similar but specific values will be different.

Result from Report for Texas

```
Report for TX
Winning Numbers: 7 21 53 58 61  14
Tickets sold: 3,784
Average prize: $0.12

Biggest Winner
Hyman Dana
Beaumont, TX 77701
2 7 21 49 53   14
Prize: $50
```

Result for Biggest Winner

```
Biggest Winner

Stephanie Goza
Camden, NJ 8102
8 17 31 38 68   7
Prize: $50.00
```

Result for Multiple Drawings

```
Number of drawings: 488

Barbara Patterson
Louisville, KY 40219
5 29 44 51 61   6
Prize: $732,500,000.00
```

Result for Big Winners ($50 or higher)

```
Big Winners ($50 or higher)

Lawanda Hughes
Harveyville, KS 66431
37 39 43 54 59   5
Prize: $50.00

Melody Crider
Ridley Park, PA 19078
37 39 45 54 68   5
Prize: $50.00

Brandon Fowler
Rosemount, MN 55068
37 39 54 71 74   5
Prize: $50.00

Mary Lehn
Rocky Mount, NC 27804
32 37 39 54 59   7
Prize: $500.00
4 winning tickets
```

Coding Style (10 pts)

Good programming practice includes writing elegant source code for the human reader. Follow the Style Guide for Python Code. For CIS 162, follow these additional guidelines:

- Comments are at the same indentation as the corresponding code
- Provide comments before most if statements and loops
- Blank lines before every comment
- Use docstrings to describe classes and functions (see 5.9)

Turn In

A professional document **is stapled** with an attractive cover page. Do not expect the lab to have a working stapler!

- Cover page - Provide a cover page that includes your name, a title, and an appropriate picture.
- Signed Pledge – The cover page must include the following signed pledge: "I pledge that this work is entirely mine, and mine alone (except for any code provided by my instructor). " In addition, provide names of any people you helped or received help from. Under no circumstances do you exchange code electronically. You are responsible for understanding and adhering to the School of CIS Guidelines for Academic Honesty.
- Programming Journal (2nd page) – Professional coders track their time in fifteen-minute increments and journal their activities. What challenges did they encounter and how did they solve the issue? What portion of the program was completed and when? Include dates and times. In addition, provide names of any people you helped or received help from.
- Source code - a printout of your elegant source code printed from PyCharm (with your name). Please do not print in dark mode or your project will be returned ungraded!

Grading Criteria

- Pass all zyBook tests (85 pts)
- Signed pledge and programming journal (5 pts)
- Attractive source code with appropriate comments to describe separate sections of the solution. All files should have the following header at the top. (10 pts)

```
'''
YOUR NAME
Brief description of the code for other programmers

I certify that this code is mine, and mine alone, in accordance with
GVSU academic honesty policy.

COMPLETION DATE
'''
```

- **zyBook Trickery.** Any code intended to directly pass a zyBook test without following the specification will result in a significant grade adjustment. For example, do not hardcode print statements to pass zyBooks tests.
- **Late Policy.** Projects are due at the START of the class period. However, you are encouraged to complete a project even if you must turn it in late. Each 24-hour period submitted late (-10 pts). Weekends and holidays are free days. Projects are not accepted more than five days late without prior instructor approval.