COMP1005/1405 – Summer 2022
Assignment # 5
Due on Friday, June 17th by 11:59 pm.

Submission Guidelines:

Submit a single zip file called A5.zip on Brightspace.

Did you zip it correctly? You must always create the zip files using the built-in, default, archiving program available with your operating system. If we cannot easily open your zip file and extract the python files from them, we cannot grade your assignment. Other file formats, such as rar, 7zip, etc., will not be accepted.

Windows: Highlight (select with ctrl-click) all of your files for submission. Right-click and select "Send to" and then "Compressed (zipped) folder". Change the name of the new folder "A5.zip".

MacOS: Highlight (select with shift-click) all of your files for submission in Finder. Right-click on one of the files and select "compress N items…" where N is the number of files you have selected. Rename the "Archive.zip" file "A5.zip".

Linux: use the zip program.

Did you submit the correct files? You are responsible for ensuring that you have submitted the correct file(s), and the submission is completed successfully. After submitting your A5.zip file to Brightspace, be sure that you download it and then unzip it to make sure that you submitted the correct files. This also checks that your zip file is not corrupted and can be unzipped.

You can submit as many times as you wish and Brightspace will save your latest submission. I would suggest that you submit as soon as you have one question done and keep re-submitting each time you add another problem (or partial problem).

We will not accept excuses like "I accidentally submitted the wrong files" or "I didn't know that the zip file was corrupt" or "My Internet was down" or "My computer was not working" after the due date.

If you are having issues with submission, contact the instructor before the due date.

Late Policy

The assignment is due on Friday, June 17th, 2022 by 11:59 PM (Ottawa time). Late submissions are allowed until Sunday, June 19th, 2022, 11:59 AM (Ottawa time) without any penalty.

Multiple submissions are allowed and Brightspace will only keep your LAST submission. The submission link will close on Monday, June 20th, 2022, 12:01 AM.

Here is a summary of the penalties based on the submission time.

Last submission Day/Time	**Penalty applied**
Friday, June 17th, 2022 by 11:59 PM (Ottawa time)	No penalty
Sunday, June 19th, 2022, 11:59 PM (Ottawa time)	No penalty
Monday, June 20th, 2022, 12:01 AM or later	100%

Regret Clause

If you think you have committed an academic violation (plagiarism), you have 12 hours after the final allowable submission time (Monday, June 20th, 2022, 12:00 PM, for this assignment) to invoke this regret clause.

If you do, you will receive ZERO for the assignment (or portions of it that you invoke the clause for) but will not receive any further sanctions from the Dean's office. Your assignment will still be used when we are checking for plagiarism, but you would not face any further sanctions if it was decided that you had committed an academic violation.

For example, if you collaborated too closely with another student and this is discovered, you would not be further sanctioned. The other student, if they did not also invoke this clause, may still be further sanctioned by the Dean's office.

This is not a group project, so collaboration is not allowed on this assignment.

Self Evaluation Quiz Criteria

SE Quiz 5 is a follow-up quiz for Assignment-5. Submit this assignment to take the quiz on Brightspace.

You will be asked some leading questions to help you assess the problem-solving skills that you might have achieved by doing this assignment, such as:

- Do you feel more comfortable with programming terminologies?
- Did you draw a flowchart or write a pseudocode to help understand an algorithm?
- Did you try to break your problem into smaller sub-problems and solve each sub-problem separately?
- Did you use examples to help understand a problem?
- Could you explain the problems in your own words to someone else in this class?
- Did you find similar or related statements that we've talked about in lecture or tutorial, and did you try to understand how the problems were similar and how they were different?
- Did you ask a question on discord that required you to communicate what you understood and what was still confusing?
- Did you answer a peer's question on discord?
- Do you feel more comfortable using Python data structures like lists and dictionaries?
- Do you feel more comfortable using strings and File I/O methods to read from, process, write to files in Python?
- Did you trace your code/algorithm and make sure it follows correctly and produce correct results?
- Did you try to identify the stages of the programming life cycle while you were solving a problem for this assignment?
- Do you feel more comfortable with identifying the errors in your code and handling those properly?
- For the given assignment, which letter grade would you give yourself (A+, …, F)?
- In 2-3 sentences, justify your letter grade selection.

Note: only the last two questions are mandatory for this quiz.

Topics: Control structures, Functions, Data Structures, Exception handling, OOP

In this last assignment, you will apply the object-oriented approach to the game you previously wrote in A2. The output of the game will remain the same but you will re-design the code so that it must use the following programming tools.

- classes and objects to represent the elements of this game
- data structures to store and manage data efficiently
- exception handling to catch and handle runtime errors
- functions to divide a large problem into smaller ones and reuse codes where necessary

You may want to use the algorithm that you developed for A2 since the conditions of the game have not changed. This assignment aims to produce a cleaner, modularized and well-designed code.

This specification provides the requirements that must be used to re-design the code for A5. Section 1 describes the classes that you should use to design the new data types. You may create additional classes and/or use additional attributes/methods in these classes. Section 2 guides you through the decomposition step to divide the game logic into smaller problems. You are required to write functions (or methods) for these smaller problems (basically re-use your previous code and put them inside those functions) and you are encouraged to use additional functions where applicable. You can select any data structures that you want to use in this assignment.

You will create and submit one py file **a5.py** for this assignment. Please be sure to read through the entire assignment before you begin to redesign your algorithm and code it. You may have to spend more time redesigning your algorithm than coding it. You are required to present your algorithm if you seek any help from the TAs or the instructor.

1. Required classes

Your program must implement the class definitions described in this section and create/use objects and methods of these classes in your program where applicable. You must include the **`__init__`** method in the class definitions for initializing their attributes.

(a) **DNA class** has at least two attributes/values: **length** (int) and **strand** (str). Add a method to this class definition that randomly generates a DNA strand of length **length**.

(b) **Player class** has at least two attributes/values: **name** (string) of the player and **score** (int). Add a method to this class definition that updates the score of the player.

Note that, it is possible to improve these class definitions. You can write additional classes/attributes/methods in your program if necessary. For example, you may want to create a separate class called Quiz that handles the tasks of greeting the player, displaying game options, asking for user input, validating player's answers, and quitting the game. You are welcome to discuss your plan with the instructor/TAs.

2. Decomposing the problem and writing functions for them

To help with the *decomposition* step, the game logic is broken down into the following smaller problems.

1. Welcome the player, ask for their name and show them the rules.
2. Ask user input for the length of the DNA, validate user input
3. Generate a random DNA strand of a predefined length
4. Display the options for choosing a quiz question.
5. Compute the complement of a DNA strand
6. Compute the reverse of a DNA strand
7. Compute the compressed form of a DNA strand
8. Compute the expanded form of a DNA strand
9. Ask quiz questions and evaluate the player's answers
10. Update player's scores
11. Determine if the player wins and show the appropriate message before terminating the game

You will write either functions or methods for most of these sub-problems. Note that, methods are just functions that are defined inside of a class. If any class definition covers one or more of these tasks, you can include them as methods in that class definition.

You will find the names and the sample output for each of these functions below. You can decide the parameter(s) and the return value(s) for the functions/methods. You may want to further divide the amount of work done in each of these functions into smaller sections and write additional functions for them. Be sure to use them properly inside of the mandatory functions so they perform the required tasks. Only the required functions will be graded.

Note that I have used sample outputs similar to that of A2 so that you can refer to your previous algorithm/code and correctly decompose that for A5. If you previously customized your output in some other way, you can follow that version.

1. Write a function named `welcome()` that performs the task of subproblem (1).

   The sample output for this function can be:
   ```
   Welcome to the DNA quiz game!
   ```

```
Win the game by scoring at least 10 points!
Enter your username >> Miki
```

2. Write a function named `get_length()` that performs the task of subproblem (2).

   The sample output for this function can be the following:

```
Hi Miki, please enter a positive integer for the DNA length >> -4
Invalid input, please try again..
Hi Miki, please enter a positive integer for the DNA length >> 4
```

3. Write a method named `generate_dna()` in the DNA class that takes an integer (length) and performs the task of subproblem (3). This method does not print anything, so there is no sample output for it.

4. Write a function named `display_options()` that performs the task of subproblem (4). The sample output for this function can be the following:

```
Select an option [1-4] to answer a question or 5 to quit the game.
1. Complement [2 points]    2. Reverse [2 points]    3. Compress [3 points]    4.
Expand [3 points]   5. Quit
```

5. Write a function named `complement()` that takes a string (DNA strand) and returns its complement. This function performs the task of subproblem (5).

   Sample outputs:

```
>>> answer = complement(GCTA)
>>> print(answer)
CGAT

>>> answer = complement(AGGA)
>>> print(answer)
TCCT
```

6. Write a function named `reverse()` that takes a string (DNA strand) and returns its reverse. This function performs the task of subproblem (6).

   Sample outputs:

```
>>> answer = reverse(CTGC)
>>> print(answer)
CGTC
```

```
>>> answer = reverse(GATT)
>>> print(answer)
TTAG
```

7. Write a function named `compress()` that takes a string (DNA strand) and returns its compressed form. This function performs the task of subproblem (7).

   Sample outputs:

   ```
   >>> answer = compress(GGCG)
   >>> print(answer)
   2GCG
   ```

   ```
   >>> answer = compress(TATA)
   >>> print(answer)
   TATA
   ```

8. Write a function named `expand()` that takes a string (DNA strand) and returns its expanded form. This function performs the task of subproblem (8).

   Sample outputs:

   ```
   >>> answer = expand(2CAG)
   >>> print(answer)
   CCAG
   ```

   ```
   >>> answer = compress(GT2C)
   >>> print(answer)
   GTCC
   ```

9. Write a method `update_score()` in the Player class that takes an int (score) and updates the player's current scores. This function performs the task of subproblem (10).

10. Write a main function that must initialize all variables and/or objects and call all other functions in the correct sequence to play the game. You may want to write additional functions/methods for subproblems 8 and 11 or you can write code for them directly in the main function. Do not forget to add the main guard to call `main()`. A complete sample run of this program is available in section 4.

3. Function definitions, parameters, and return values

- Note that, you can decide the input parameters and the return values for each function.

- You must not use any global variables in this program but you can use global constants if needed. Recall that a global variable is declared outside of any function on the top level of the program that is available to the entire program and its value can be changed. Global variables should always be avoided if possible. Whereas, a global constant is also declared outside of any function on the top level of the program that is available to the entire program but its value must not be changed.

- You can use a simple rule to determine the parameters/return values for any function. If you need to access some data (variable/data structure/object) inside of a function, instead of using a global variable you must pass that as a parameter to the function. Similarly, if your function creates or updates some value/data structure/object that you want to make available to some other parts of your program, your function should return that. In this way, the calling function should be able to save and use the returned value(s).

- Please make sure that your program does not crash on valid inputs (valid inputs will not cause runtime errors, so it is not about exception handling). Otherwise, the TAs will have the discretion to decide if they want to debug your code and continue grading, or just give zero to that part of your code.

- Marks distribution will be based on both the correctness of the output and the efficient use of the programming tools. So, if your program produces the correct output but does not use any of the required programming tools (that means your A5 is essentially the same as A2), it will not receive more than 50% of the total marks.

- Add brief docstrings to your functions/classes so that the grader knows how your functions/classes should behave.

- Use exception handling to handle all possible run-time errors so your functions do not crash on invalid inputs.

4. A complete sample output

A sample run of your a5.py may look like this.

```
Welcome to the DNA quiz game!
Win the game by scoring at least 10 points!
Enter your username >> Miki
Hi Miki, please enter a positive integer for the DNA length >> 5

Select an option [1-4] to answer a question or 5 to quit the game.

1. Complement [2 points]   2. Reverse [2 points]   3. Compress [3 points]   4. Expand
[3 points]   5. Quit
> 2
What is the reverse of TGAAA? aaaGt
Great!! Total point: 2
Play again?

1. Complement [2 points]   2. Reverse [2 points]   3. Compress [3 points]   4. Expand
[3 points]   5. Quit
> 1
What is the complement of GGGGA? ccccT
Good job, your total score is: 4
Play again?

1. Complement [2 points]   2. Reverse [2 points]   3. Compress [3 points]   4. Expand
[3 points]   5. Quit
> 1
What is the complement of CCTGA? cctgg
That was wrong! The correct answer is GGACT. You still have 4 points.
Play again?

1. Complement [2 points]   2. Reverse [2 points]   3. Compress [3 points]   4. Expand
[3 points]   5. Quit
> 3
What is the compressed form of GCCTG? g2cTg
That was correct! Total point so far: 7
Play again?

1. Complement [2 points]   2. Reverse [2 points]   3. Compress [3 points]   4. Expand
[3 points]   5. Quit
> 4
What is the expanded form of AG2AG? aaa
That was wrong! The correct answer is AGAAG. Your score so far 7.
Play again?

1. Complement [2 points]   2. Reverse [2 points]   3. Compress [3 points]   4. Expand
[3 points]   5. Quit
> 4
What is the expanded form of C2GCA? cggca
Awesome! Your score is: 10
Play again?

1. Complement [2 points]   2. Reverse [2 points]   3. Compress [3 points]   4. Expand
[3 points]   5. Quit
```

```
> 2
What is the reverse of TGCGG? ggcgt
Great!! Total point: 12
Play again?

1. Complement [2 points]   2. Reverse [2 points]   3. Compress [3 points]   4. Expand
[3 points]   5. Quit
> 5
Great job Miki, you won the game with 12 points! Congrats!
```

Invalid submissions that may incur penalties

- Submissions with an incorrect file name or format
- Missing name and ID
- Submissions that crash (i.e., terminate with an error) on execution may receive a mark of zero (0).
- Others: functions without docstrings, use of global variables, missing one or more required components, not using input validation to repeat prompts where necessary, use of external modules not discussed in class.

A5.zip Submission Recap

You must submit the python script **a5.py** with the A5.zip file.

Marking scheme

The marking scheme will be posted later on Discord and Brighspace.