Project 0 - Description
(Building a Complex Number Object)

**Project Overview:**
For this programming assignment, we will be learning how to build data structures from formal specifications. In particular, we will be building a class for complex numbers. By default, python has no complex number class. In order to do things with complex numbers, we would either have to make our own class for it or import another library that has the class.

Complex numbers are numbers that take the form (a + bi) where a and b are real numbers. We will be implementing a class that contains a complex number as well as overwriting some of the arithmetic operators so that we can perform basic operations such as addition, subtraction, multiplication, and division. We will also be practicing some of the techniques necessary to succeed in this course.

**Core Tasks:**
1. Write the Complex class.

---

**Program Requirements:**
**Task 1:** Write the Complex class.

For task 1, you will need to write a Complex class. To receive any credit, your class must follow the specifications below exactly:

- **Class Name:** Complex
- **Methods:**
  - `__init__(<Complex> self, <int\float> a, <int\float> b) → <Nonetype> None`
    - Complexity: O(1)
    - **Valid Input:** An integer or floating point number from [-∞, ∞].
    - **Error Handling:** Raises a `ComplexConstructorTypeError` if a or b are given invalid input.
    - **Instance variables:**
      - `<int\float>` a
        - Note: This variable contains the “a” part of the complex number (a + bi).
      - `<int\float>` b
        - Note: This variable contains the “b” part of the complex number (a + bi).
      - Any other instance variables you want.
  - `__add__(<Complex> self, <Complex> other) → <Complex> returnValue`
- **Complexity:** O(1)
- **Valid Input:** A Complex object.
- **Error Handling:** Raises a `TypeError` exception if the `__add__` method is called with something other than a complex number.
- **Note:** This is the add method. It will add two complex numbers together and return a new complex number containing their summation.
  - Formula: \((a + bi) + (c + di) = (a + c) + (b + d)i\)
- `__sub__`(Complex `self`, Complex `other`) → Complex `returnValue`
  - **Complexity:** O(1)
  - **Valid Input:** A Complex object.
  - **Error Handling:** Raises a `TypeError` exception if the `__sub__` method is called with something other than a complex number.
  - **Note:** This is the subtract method. It will subtract one complex number from the other and return a new complex number containing their difference.
    - Formula: \((a + bi) - (c + di) = (a - c) + (b - d)i\)
- `__mul__`(Complex `self`, Complex `other`) → Complex `returnValue`
  - **Complexity:** O(1)
  - **Valid Input:** A Complex object.
  - **Error Handling:** Raises a `TypeError` exception if the `__mul__` method is called with something other than a complex number.
  - **Note:** This is the multiplication method. It will multiply two complex numbers and return a new complex number.
    - Formula: \((a + bi) \times (c + di) = (ac - bd) + (bc + ad)i\)
- `__truediv__`(Complex `self`, Complex `other`) → Complex `returnValue`
  - **Complexity:** O(1)
  - **Valid Input:** A Complex object.
  - **Error Handling:** Raises a `TypeError` exception if the `__truediv__` method is called with something other than a complex number.
  - **Note:** This is the division method. It will compute the division of two complex numbers and return a new complex number.
    - Formula: 
      \[
      \frac{(a + ci)}{(b + di)} = \frac{(ac + bd)}{(c^2 + d^2)} + \frac{(bc - ad)}{(c^2 + d^2)}i
      \]
`__str__(<Complex> self) → <str> return_value`

- **Complexity:** O(1)
- **Valid Input:** None.
- **Note:** This is the string method method. It will return the string representation of the complex number.
  - Format: “(a + bi)”
  - Replace the a/b with the value from the corresponding instance variable.

---

**Remarks:**

1. All methods in the complex class have a hard complexity upper bound of O(1). Your implementation **must** be in O(1).
2. For this assignment, you **must** make your own complex number class using basic python. Using the built-in complex class or any other data structure will result in a 0 for the assignment.
3. **Warning:** You are not allowed to import anything into the source file with the classes. The only exception is the math library if you want to use things like cos(). All programs must be written in basic Python 3.5+. You may import your class into your main function to test it.
4. You will need to make your own main for testing your code using the methodology described in the lab. Testing your own code is a very common practice in the industry. A starter test class will be provided (see `p0_test.py`).
5. **Warning:** I will be testing your code with a more robust main that will check all of the corner cases and uses a wider variety of objects. Keep this in mind while developing your data structures. You will need to use the methodology discussed in the lab to figure out all corner cases and account for them. Your programs **must** be robust (i.e. does not crash when given bad input or told to peak/front when the stack/queue is empty etc.)
6. All programming assignments are to be done **individually.** Your code will be looked at with professional software for cheating. **Warning:** This includes using online sources. Be extra careful with your code. Do not ever show your work to anyone other than the TA (me) or the professor. They will most likely copy your work and you will both fail.

**Submission Requirements:**

In order to receive any credit for the assignment the student **must** do the following:

1. Name your program “<Duck-ID>_p0.py”. (i.e. my duck ID is jhall10 so my submission would be named `jhall10_p0.py`. **Note:** your duck-ID is the same as your email id and the username to log on to CIS computers **not** your 951… number that is your UO PID.
2. Submit **only** the python file onto Canvas.

That’s it! Make sure that you test your code on the terminal to make sure it works.
Grading:
Your work will be graded along three primary metrics: Correctness, Completeness, and Elegance.

Correctness: (60% of total grade)
- Your program performs the operations according to the formula provided.
- Your program follows the given I/O specifications.
- Your program is robust and properly handles invalid input.

Completeness (25% of total grade)
- You completed the class according to specifications and it does not have any syntax errors.

Elegance: (15% of total grade)
- Your program is well organized.
- You make good use of whitespace, comments, and the doc strings.
- You write your code in a readable manner.
- You use descriptive variable/function/class names.

Late Policy:
The late policy for this class is as follows. Make sure to read the info below carefully:
Your homework is always due on a Wednesday at 11:59 pm. On Thursday, I will take 10% off of the total points available. On Friday, I will take 20% off. No homework will be accepted after Friday. Keep this in mind when submitting your assignment: It may be better for you to seek help and submit the homework a day late than to submit on time and fail.

If you encounter an unfortunate event or are working with a disability: Please email or speak to me. I am super flexible and am always on your side. I will give extensions as needed and am willing to work with you to make sure you get the most out of this course.