

Hints for the Python Revel Quizzes and Programming Projects

For Introduction to Python and Data Structures 2E by Y. Daniel Liang

If you see any errors or have comments or suggestions, please email me at y.daniel.liang@gmail.com.

Please check this page often. We update this page frequently in response to student questions. Last update: 2/22/2020.

NOTE: The quiz is a third-party product. Many end-of-section quizzes use a different naming convention for variables and functions/methods. For example, it uses `number_of_credits` to name a variable and `get_radius()` to name a getter method. But it would be named `numberOfCredits` and `getRadius()` if the naming convention of this book is used.

Hints for End-of-Section Assignments (Programming Quizzes):

Chapter 4 Quiz 4.3 #3:

The newline character is covered in Section 4.3.3. Use `'\n'` for the newline character.

Chapter 4 Quiz 4.3 #4:

The tab character is covered in Section 4.3.3. Use `'\t'` for the tab character.

Hints for End-of-Chapter Assignments (Programming Projects):

Chapter 1: Programming Project 1: Exercise01_01

Hint:

Please check your spelling of the words carefully. It is case sensitive. The words are separated by exactly one space.

Chapter 1: Programming Project 3: Exercise01_11

Hint:

In one year, the population will be

$312032486 + 365 * 24 * 60 * 60 / 7 - 365 * 24 * 60 * 60 / 13 + 365 * 24 * 60 * 60 / 45$

In two years, the population will be

$312032486 + 2 * 365 * 24 * 60 * 60 / 7 - 2 * 365 * 24 * 60 * 60 / 13 + 2 * 365 * 24 * 60 * 60 / 45$

You can compute the population in three years, four years, and five years.

Chapter 2: Programming Project 1: Exercise02_05

How would you write this program? Here are some hints:

Step 1: Prompt the user to enter subtotal (float) and gratuity (float) rate into variables `subtotal` and `gratuityRate`. Note the `gratuityRate` is in percent. For example, if it is 15%, you enter 15 as the input for `gratuityRate`.

Step 2: Compute gratuity: `gratuity = subtotal * gratuityRate / 100`.

Step 3: Compute total: `total = subtotal + gratuity`.

Step 4: Display gratuity and total. Note you need to first display gratuity and then total. For all the programming projects, please pay attention to output order.

Chapter 2: Programming Project 2: Exercise02_07

Hint: You must use integers in this program. Use

```
int(input("Enter the number of minutes: "))
```

to obtain the input. See LiveExample 2.4 in Section 2.8.2 for reference.

Now, the key to solve this problem is to use the correct math.

Step 1: Get the totalNumberOfDays from the minutes (i.e., $\text{totalNumberOfDays} = \text{minutes} // (24 * 60)$).

Step 2: Get the numberOfYears from the numberOfDays (i.e., $\text{numberOfYears} = \text{numberOfDays} // 365$).

Step 3: Get the remainingNumberOfDays from totalNumberOfDays (i.e., $\text{remainingNumberOfDays} = \text{totalNumberOfDays} \% 365$).

Chapter 2: Programming Project 3: Exercise02_13

Use integer in this program.

Hint: The last digit of number is $\text{number} \% 10$. Discard last digit is number using $\text{number} // 10$.

Chapter 2: Programming Project 4: Exercise02_04Extra

Use float numbers in this program.

Chapter 2: Programming Project 5: Exercise02_19

Hint: Prompt the user to enter investmentAmount (float), annualInterestRate (float), and numberOfYears (int), and apply the formula to compute the accumulatedValue.

The annualInterestRate is entered in percentage. In the formula, you need to use monthlyInterestRate, which is $\text{annualInterestRate} / 1200$. See LiveExample 2.8 in Section 2.14 for reference.

Chapter 3: Programming Project 1: Exercise03_01

There are typos in the description. For the correct description, please see Exercise 3.1 in the Programming Exercise from the Book section in Chapter 3 from the TOC.

How would you write this program? Here are some hints:

Step 1: Prompt the user to enter a, b, c. All are float numbers.

Step 2: Compute discriminant.

Step 3: if discriminant < 0, display an error message.

Step 4: elif discriminant == 0, compute and display one root.

Step 5: else compute and display two roots.

Common errors from this project: In Step 1, your code should read in float values. In Step 2, discriminant is $\text{discriminant} = b * b - 4 * a * c$. In Step 5, please note that root1 is $r1 = (-b + \text{discriminant} ** 0.5) / (2 * a)$. Another common mistake is to compute the roots before the if statement. Please compute roots in Step 5.

Chapter 3: Programming Project 2: Exercise03_03

How would you write this program? Here are some hints:

Step 1: Prompt the user to enter a, b, c, d, e, f. All are float numbers.

Step 2: Compute $\text{detA} = a * d - b * c$.

Step 3: if $\text{detA} == 0$, display an error message.

Step 4: else compute x and y, and display the result.

Common errors from this project: A common mistake is to compute x and y before the if statement. Please compute x and y in Step 4.

Chapter 3: Programming Project 3: Exercise03_11

Prompt the user to enter year (int) and test if year is a leap year.

Hint: To test if a year is a leap year, see Section 3.11.

Chapter 3: Programming Project 4: Exercise03_21

Hint: Prompt the user to enter year, month, and dayOfMonth. Note that all input are integers. You need to adjust year and month as follows:

if month is 1, set month = 13 and year = year - 1.

if month is 2, set month = 14 and year = year - 1.

You can now use Zeller's formula to compute h:

$$h = \left(q + \frac{26(m+1)}{10} + k + \frac{k}{4} + \frac{j}{4} + 5j \right) \% 7$$

Chapter 3: Programming Project 5: Exercise03_23

All the input are float numbers.

Hint: A point (x, y) is in the rectangle centered at (0, 0) with width 10 and height 5 if $|x| \leq 5$ and $|y| \leq 2.5$. That is, $x \leq 5$ and $x \geq -5$ and $y \leq 2.5$ and $y \geq -2.5$.

Chapter 4: Programming Project 1: Exercise04_55

Hint: Prompt the user to enter the numberOfSides (int) and the lengthOfSide (float) of a polygon, and apply the formula to compute the area of the polygon.

Chapter 4: Programming Project 2: Exercise04_23

Hint: Prompt the user to enter a character for grade, and test if it is A/a, B/b, C/c, D/d, F/f and display its corresponding value, otherwise displays “invalid grade”.

Chapter 4: Programming Project 3: Exercise04_25

Hint: Prompt the user to enter year (int) and month (str), and displays the number of days in the month. To test if a year is a leap year, see Section 3.11.

Chapter 4: Programming Project 4: Exercise04_29

How would you write this program? Here are some hints:

Step 1: Prompt the user to enter the first 9-digit of an ISBN number as a string into s.

Step 2: Write a if/else statement. If len(s) is not 9 or s.isdigit() is False, display "Incorrect input".

Step 3: else compute the checksum as follows:

Step 3.1: Compute checksum = (d1 * 1 + d2 * 2 + d3 * 3 + d4 * 4 + d5 * 5 + d6 * 6 + d7 * 7 + d8 * 8 + d9 * 9) % 11

Note d1 = ord(s[0]) – ord('0') and d2 = ord(s[1]) – ord('0'), etc.

Step 3.2: Use an if/else statement to get the checksum character: if checksum is 10, the character is X, otherwise it is checksum itself.

Chapter 4: Programming Project 5: Exercise04_31

Hint: Read a hex character into hex. Convert it into an uppercase using hex.upper(). Test if hex is '0', ..., '9', 'A', ..., 'F' to display the corresponding binary number.

Chapter 5: Programming Project 1: Exercise05_01

How would you write this program? Here are some hints:

Step 1: Initialize variables. Your program needs to count the number of positives, number of negatives, and total. Think about what initial values are appropriate for these variables. Note that the average can be computed using total / (numberOfPositives + numberOfNegatives).

Step 2: Prompt the user to read a number into variable number.

Step 3: Write a while loop to do the following:

Step 3.1: the loop continuation condition is (number != 0)

Step 3.2: If number > 0, increase numberOfPositives by 1; if number < 0, increase numberOfNegatives by 1.

Step 3.3: Add number to total.

Step 3.4: Prompt the user to a new input and assign it to number.

Step 4: After the loop is finished, if (numberOfPositives + numberOfNegatives == 0), this means “No numbers are entered except 0”. Your program should display exactly this message. Otherwise, display the number of positives, negatives, and average. To compute average, use total / (numberOfPositives + numberOfNegatives).

Chapter 5: Programming Project 2: Exercise05_09

How would you write this program? Here are some hints:

First, I suggest that you to read Section 5.9.2. This section will give you half of the solution.

Step 1: Note that the current year tuition is 10000. In one year (same as to say after the first year), the tuition will be $\text{tuition} += \text{tuition} * 1.05$ (10500). Compute the tuition in 10 years using a loop and save the result in variable `tuition`. This will be the tuition in 10 years. Check your result. It should be 16288.95.
Step 2: Now compute the tuition after the 11th year, 12th year, 13th year. The tuition after the 10th year is 16288.95. The sum of the tuition after the 10th, 11th, 12th, and 13th is the total cost of four years tuition in ten years (same as to say after the 10th year).

Chapter 5: Programming Project 3: Exercise05_11

Note that we assume that the number of students is at least 2. So, your program does not need to consider the case for one or no students.

How would you write this program? Here are some hints:

Step 1: Prompt the user to read the number of the students into variable `numberOfStudents` (int).
Step 2: Prompt the user to read first student name into `student1` (str).
Step 3: Prompt the user to read first student score into `score1` (float).
Step 4: Prompt the user to read second student name into `student2` (str).
Step 5: Prompt the user to read second student score into `score2` (float).
Step 6: If `score1 < score2`, swap `student1` with `student2` and `score1` with `score2`. Throughout the program, we will use `student1` and `score1` to store the student name and score with the highest score and `student2` and `score1` to store the student name and score with the second highest score.
Step 7: Now write a loop to read the next student name and score. Consider the following cases:
Step 7.1: if `score > score1`, assign `score1` to `score2` and `score` to `score1`, `student1` to `student2` and `name` to `student`.
Step 7.2: else if `score1 > score2`, assign `score` to `score2` and `name` to `student2`.
Step 8: Display the top two students' name and score.

Chapter 5: Programming Project 4: Exercise05_23

For this program, you need to use the formula for computing `monthlyPayment` and `totalPayment` in Section 2.14.

How would you write this program? Here are some hints:

Step 1: Prompt the user to read `loanAmount` (float) and `numberOfYears` (int).
Step 2: Print the header using the format function, "Interest Rate", "Monthly Payment", "Total Payment";
Step 3: Write a while loop as follows:
Step 3.1: Before the loop, Initialize `annualInterestRate` to 5.0.
Step 3.2: The loop continuation condition is `<= 8.0`.
Step 3.3: In the loop body, compute `monthlyPayment` and `totalPayment` using the formula. Note the `monthlyInterestRate` is `annualInterestRate / 1200`. Display `annualInterestRate`, `monthlyPayment`, and `totalPayment` in one line using the format function to format the output. **Please note: in the first column, you need to put three digits after the decimal point. For example, 5.000% is correct, but 5% is wrong.**
Step 3.4: Don't forget: `annualInterestRate += 1.0 / 8` in the loop.

Chapter 5: Programming Project 5: Exercise05_59

How would you write this program? Here are some hints:

Step 1: Initialize variables `numberOfVowels` and `numberOfConsonants` with 0.
Step 2: Prompt the user to enter a string.
Step 3: For each character in the string, do the following:
Step 3.1: Convert the character to uppercase.
Step 3.2: If the character is 'A', 'E', 'I', 'O', or 'U', increase `numberOfVowels` by 1.
Step 3.3: else if the character is a letter, increase `numberOfConsonants` by 1.
Step 4: Display the result using the wording as shown in the sample output.

Chapter 5: Programming Project 6: Exercise05_57

How would you write this program? Here are some hints:

Step 1: Prompt the user to enter the first 12 digits of an ISBN-13 into a string `s`.
Step 2: If the length of the input is not 12, exit the program using `sys.exit()`.

Step 3: Initialize variable sum. (In Step 4, we will compute sum. sum will be $d1 + 3*d2 + d3 + 3*d4 + \dots + d11 + 3*d12$. Note d1 is s[0], d2 is s[1], etc.)

Step 4: for each i from 0 to len(s) – 1, do the following:

Step 4.1: If (i % 2 is 0), add (ord(s[i]) - ord('0')) to sum.

Step 4.2: else, add (ord(s[i]) - ord('0')) * 3 to sum.

Step 5: Obtain checksum that is $10 - \text{sum} \% 10$.

Step 6: Display the entire ISBN-13 string whose last digit is checksum. Note that if checksum is 10, display digit 0.

Chapter 6: Programming Project 1: Exercise06_13

How would you write this program? Here are some hints:

The program has two functions: the main function and the m(i) function.

Step 1: Implement the m(i) function to return the summation as shown in the formula for m(i) given in the description.

Step 1.1: Initialize sum with 0.

Step 1.2: for each k from 1 to i, add $k / (k + 1)$ to sum.

Step 1.3: return sum.

Step 2: Implement the main function as follows:

Step 2.1: Display the header of the table using the format function. The header is “i m(i)”.

Step 2.2: Write a for loop as follows: for each i from 1 to 20, display i and the result from invoking m(i). The result of m(i) is displayed with four digits after the decimal point. So, 0.5 should be displayed 0.5000 using the format function.

Chapter 6: Programming Project 2: Exercise06_07

How would you write this program? Here are some hints:

The program has two functions: the main function and the futureInvestmentValue function.

Step 1: Implement the futureInvestmentValue function to compute the futureInvestment value from investmentAmount, monthlyInterestRate and years using the formula from Exercise 2.19 in Chapter 2. This function return futureInvestmentValue.

Step 2: Implement the main function as follows:

Step 2.1: Prompt the user to enter investmentAmount (float) and annualInterestRate (float).

Step 2.2: Write a for loop as follows: for each year from 1 to 30, display year and the result from invoking futureInvestmentValue(investmentAmount, annualInterestRate / 1200, year). You need to use the format function to display formatted output.

Chapter 6: Programming Project 3: Exercise06_53

How would you write this program? Here are some hints:

The program has two functions: the main function and the count(s, a) function.

Step 1: Implement the count(s, a) as follows:

Step 1.1: Initialize count.

Step 1.2: for each i from 1 to len(s) - 1, if s[i] == a, increase count by 1.

Step 1.3: return count.

Step 2: Implement the main function as follows:

Step 2.1: Prompt the user to enter a string s.

Step 2.2: Prompt the user to enter a char.

Step 2.3: Simply invoke count(s, ch) to return the count and display the result as shown in the sample run.

Chapter 6: Programming Project 4: Exercise06_23

First, please read Listing 2.7 ShowCurrentTime.java in Section 2.13. This is very helpful for this exercise.

How would you write this program? Here are some hints:

The program has two functions: the main function and the convertMillis(long millis) function.

Step 1: Implement the convertMillis(millis) function as follows:

Step 1.1: Obtain seconds from millis.

Step 1.2: Obtain totalMinutes from seconds.

Step 1.3: Obtain minutes from totalMinutes % 60.

Step 1.4: Obtain totalHours from totalMinutes // 60.

Step 1.5: Return a string: hours + ":" + minutes + ":" + seconds.

Step 2: Implement the main function as follows:

Step 2.1: Prompt the user to enter an integer into variable totalMillis.

Step 2.2: Invoke convertMillis(totalMillis) to return a string.

Step 2.3: Display this string.

Chapter 6: Programming Project 5: Exercise06_02Extra

As always, there are many ways to solve a problem. An easy way might be first convert the binary to decimal and then convert the decimal to hex. Converting a decimal to hex is covered in Section 5.9.3. Converting a binary number to decimal is similar to converting a hex to decimal, which is covered in Section 6.12.

Chapter 7: Programming Project 1: Exercise07_01

How would you write this program? Here are some hints:

Step 1: Enter the scores in one line separated by spaces.

Step 2: Split the line into a list items and get scores from the items using list comprehension. The scores are float numbers.

Step 3: Initialize variable best to keep the best score. Set the initial value to 0.

Step 4: For each score: compare it with best. If it is greater than best, assign it to best.

Step 5: For each score, compare it with best, assign the grade for the student.

Chapter 7: Programming Project 2: Exercise07_03

How would you write this program? Here are some hints:

Step 1: Enter the numbers in one line separated by spaces.

Step 2: Split the line into a list items and get numbers from the items using list comprehension. Numbers are integers.

Step 3: Create list counts with 100 elements. The initial values in counts are 0.

Step 4: For each value in numbers, update counts.

Step 5: For each value in counts, display the value and its count. Note the 0 count is not displayed. Single count is displayed in word “time”, and multiple count is displayed in word “times”.

Chapter 7: Programming Project 3: Exercise07_05

How would you write this program? Here are some hints:

Step 1: Enter the numbers in one line separated by spaces.

Step 2: Split the line into a list list1. Numbers are integers.

Step 3: Create a new empty list list2.

Step 4: For each number in list1, if it is not in list2, add to list2.

Step 5: list2 now contains the distinct numbers from list1.

Chapter 7: Programming Project 4: Exercise07_09

How would you write this program? Here are some hints:

Step 1: Implement the mean(x) function as follows: return sum(x) / len(x)

Step 2: Implement the deviation(xe) function as follows:

Step 2.1: Initialize squareSum.

Step 2.2: Write a loop. For each value in x, add (value – mean(x)) ^ 2 to squareSum.

Step 2.3: return sqrt(squareSum / (x.length – 1))

Step 3: Implement the main function as follows:

Step 3.1: Enter the numbers in one line separated by spaces. Numbers are float numbers.

Step 3.2: Split the line into a list and get numbers using list comprehension.

Step 3.3: Invoke mean(numbers) and deviation(numbers) to obtain mean and deviation for numbers.

Chapter 7: Programming Project 5: Exercise07_15

How would you write this program? Here are some hints:

Step 1: Implement the isSorted(numbers) function as follows:

Step 1.1: Write a for loop: for i from 0 to len(numbers) – 1, if (numbers[i] > numbers[i + 1]), return false.

Step 1.2: If nothing is return in the for loop, return true after the for loop.

Step 2: Implement the main function as follows:

Step 2.1: Enter the numbers in one line separated by spaces. Numbers are float numbers.

Step 2.2: Split the line into a list numbers.

Step 3: Invoke `isSorted(numbers)` to test if the elements in list are sorted.

Chapter 7: Programming Project 6: Exercise07_17

How would you write this program? Here are some hints:

Step 1: Implement the `main()` function as follows:

Step 1.1: Read string `s1`.

Step 1.2: Read string `s2`.

Step 1.3: Invoke `isAnagram(s1, s2)` to test if `s1` and `s2` are anagrams.

Step 2: Implement the `isAnagram(s1, s2)` function as follows:

Step 2.1: If `s1` and `s2` have different sizes, return false.

Step 2.2: Sort `s1` and `s2` and return true if they are the same.

Step 3: Implement the `sort(s)` function to sort a string as follows:

Step 3.1: Obtain a list from `s` simply using `list(s)`.

Step 3.2: Sort the list.

Step 3.3: Create a string from the list and return the string.

Chapter 8: Programming Project 1: Exercise08_01

How would you write this program? Here are some hints:

Step 1: Implement the `sumColumn(m, columnIndex)` function as follows:

Step 1.1: Initialize sum.

Step 1.2: Write a for loop: for `i` from 0 to 3 - 1, add `m[i][columnIndex]` to sum. Note that the row size is 3.

Step 1.3: Return sum.

Step 2: Implement the main function as follows:

Step 2.1: Create a list `m = []`.

Step 2.2: Write a for loop to prompt the user to enter a row as a string. Extract the numbers (float) in the row to create a list. Add this list to `m`.

Step 2.3: Write a for loop. For each `j` from 0 to 4 - 1, invoke `sumColumn(m, j)` and display it. Note that the column size is 4.

Hint: Make sure you use the correct row and column size. The matrix has 3 rows and 4 columns.

Chapter 8: Programming Project 2: Exercise08_05

How would you write this program? Here are some hints:

Step 1: Implement the `addMatrix(m1, m2)` function as follows:

Step 1.1: Create a two-dimensional list `m3` of the same size as `m1`.

Step 1.2: Write a nested for loop to assign `m1[i][j] + m2[i][j]` to `m3[i][j]`.

Step 1.3: Return `m3`.

Step 2: Implement the `printResult(m1, m2, m3, op)` function as follows:

Step 2.1: For each row `i` from `len(m1)`, display a row in `m1`, `m2`, and `m3`. In the row middle, display the `op` between `m1` and `m2` and display the `=` symbol between `m2` and `m3`.

Step 3: Implement the main function as follows: (numbers are float)

Step 3.1: Create list `m1`. Enter input for `m1` in a loop one row at a time.

Step 3.2: Create list `m2`. Enter input for `m2` in a loop one row at a time.

Step 3.3: Create list `m3`. Invoke `m3 = addMatrix(m1, m2)`.

Step 3.4: Display the result by invoking `printResult(m1, m2, m3)`

Chapter 8: Programming Project 6: Exercise08_35

How would you write this program? Here are some hints:

Step 1: Implement the main function.

Step 1.1: Prompt the user to enter the coordinates for the cities. `cities` is a two-dimensional list. Numbers are float.

Step 1.2: Use list comprehension to obtain a two-dimensional list for cities. Each list in `cities` is a pair of coordinates.

Step 1.3: Initialize `minTotal` and `minIndex` to store the minimum total distance and the index of the minimum total distance city. `minTotal` is `totalDistance(cities, 0)` and `minIndex` is 0.

Step 1.4: For every city with index `i`, invoke `totalDistance(cities, i)` to return the `totalDistance`. If it is < `minTotal`, assign `totalDistance` to `minTotal` and `i` to `minIndex`.

Step 1.5: Display the (cities[minIndex][0], cities[minIndex][1]) and minTotal for the central city.
Step 2: Implement distance(c1, c2). This function returns the distance between (c1[0], c1[1]) and (c2[0], c2[1]).
Step 3: Implement and totalDistance(cities, i). This function returns the total distance from city i to all other cities.

Chapter 9: Programming Project 1: Exercise09_01

Please note that you have to put all code in one file in order to submit to REVEL. The outline of the program may look like this:

```
# Exercise09_01
class Rectangle:
    # Write your code

def main():
    # Write your code
main()
```

Chapter 9: Programming Project 2: Exercise09_05

Please note that you have to put all code in one file in order to submit to REVEL. The outline of the program may look like this:

```
# Exercise09_05
import math

class RegularPolygon:
    # Write your code

def main():
    # Write your code

main()
```

Chapter 9: Programming Project 3: Exercise09_11

Please note that you have to put all code in one file in order to submit to REVEL. The outline of the program may look like this:

```
# Exercise09_11
class Point:
    # Write your code

def main():
    # Write your code

main()
```

Chapter 9: Programming Project 4: Exercise09_15

Please note that you have to put all code in one file in order to submit to REVEL. The outline of the program may look like this:

```
# Exercise09_15
```



```

class Complex:
    def __init__(self, a = 0, b = 0):
        # Implement it

    def getA(self):
        # Implement it

    def getB(self):
        # Implement it

    def __add__(self, secondComplex):
        # Implement it

    def __sub__(self, secondComplex):
        # Implement it

    def __mul__(self, secondComplex):
        # Implement it

    def __truediv__(self, secondComplex):
        # Implement it

    def __abs__(self):
        # Implement it

    def __str__(self):
        # Implement it

    def getRealPart(self):
        # Implement it

    def getImaginaryPart(self):
        # Implement it

def main():
    a = float(input("Enter the real part of the first complex number: "))
    b = float(input("Enter the imaginary part of the first complex number: "))
    c1 = Complex(a, b)

    c = float(input("Enter the real part of the second complex number: "))
    d = float(input("Enter the imaginary part of the second complex number: "))
    c2 = Complex(c, d)

    print(c1, "+", c2, "=", c1 + c2)
    print(c1, "-", c2, "=", c1 - c2)
    print(c1, "*", c2, "=", c1 * c2)
    print(c1, "/", c2, "=", c1 / c2)
    print("|" + str(c1) + "|" = " + str(abs(c1)))

```

```
main()
```

Chapter 12: Programming Project 1: Exercise12_01

Please note that you have to put all code in one file in order to submit to REVEL. The outline of the program may look like this:

```
# Exercise12_01
import math

def main():
    # Write your code

class GeometricObject:
    # Copy the code from the book

class Triangle(GeometricObject):
    # Write your code

main()
```

Chapter 12: Programming Project 2: Exercise12_25

Please note that you have to put all code in one file in order to submit to REVEL. The outline of the program may look like this:

```
# Exercise12_25
class MyList(list):
    # Write your code

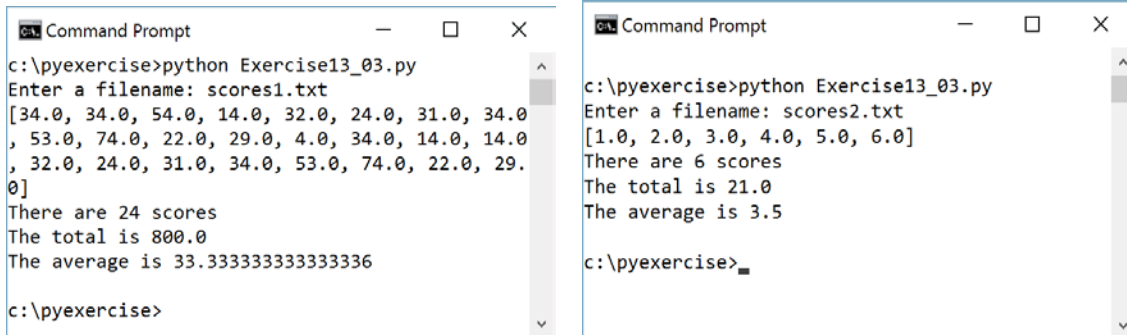
def main():
    list1 = MyList()
    list1.append("Chicago")
    list1.append("Detroit")
    list1.append("Denver")
    list1.append("Chicago")
    list1.append("Atlanta")
    list1.append("New York")
    list1.append("Seattle")
    list1.append("Dallas")
    list1.append("Atlanta")
    list1.append("New York")

    print(list1)
    index = int(input("Enter index: "))
    element = input("Enter element: ")
    print("The index of element", element, "after index", index,
          "is", list1.indexOf(element, index))
    print("The index of last element", element, "before index", index,
          "is", list1.lastIndexOf(element, index))
```

```
main()
```

Chapter 13: Programming Project 1: Exercise13_03

Due to my server security restriction, the user is not allowed to read/write files on the server. You cannot test Exercise13_03 from CheckExercise. However, you can test your code using scores1.txt and scores2.txt. Download these two files and save them as scores1.txt and scores2.txt in the same directory of your .py file. If your output matches the following two screen shots (**character by character**), your code is correct. You can then submit it to REVEL.



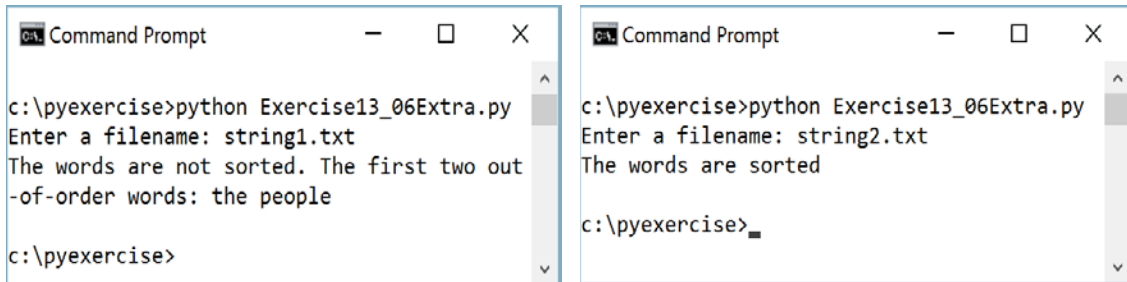
```
c:\pyexercise>python Exercise13_03.py
Enter a filename: scores1.txt
[34.0, 34.0, 54.0, 14.0, 32.0, 24.0, 31.0, 34.0, 53.0, 74.0, 22.0, 29.0, 4.0, 34.0, 14.0, 14.0, 32.0, 24.0, 31.0, 34.0, 53.0, 74.0, 22.0, 29.0]
There are 24 scores
The total is 800.0
The average is 33.333333333333336
c:\pyexercise>

c:\pyexercise>python Exercise13_03.py
Enter a filename: scores2.txt
[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
There are 6 scores
The total is 21.0
The average is 3.5
c:\pyexercise>
```

Note: Use `s.split()` to split the numbers in a string separated by whitespace characters. Don't use `s.split('')`. If `s = "1 2 3\n"`, `s.split()` will `['1', '2', '3']`, but `s.split('')` will be `['1', '2', '3\n']`.

Chapter 13: Programming Project 2: Exercise13_06Extra

Due to my server security restriction, the user is not allowed to read/write files on the server. You cannot test Exercise13_06Extra from CheckExercise. However, you can test your code using string1.txt and string2.txt. Download these two files and save them as scores1.txt and scores2.txt in the same directory of your .py file. If your output matches the following two screen shots (**character by character**), your code is correct. You can then submit it to REVEL.



```
c:\pyexercise>python Exercise13_06Extra.py
Enter a filename: string1.txt
The words are not sorted. The first two out-of-order words: the people
c:\pyexercise>

c:\pyexercise>python Exercise13_06Extra.py
Enter a filename: string2.txt
The words are sorted
c:\pyexercise>
```

Chapter 14: Programming Project 3: Exercise14_05Extra

Here are Python keywords from Appendix A.

```
keyWords = {"and", "del", "from", "not", "while",
            "as", "elif", "global", "or", "with",
            "assert", "else", "if", "pass", "yield",
            "break", "except", "import", "print",
            "class", "exec", "in", "raise",
            "continue", "finally", "is", "return",
            "def", "for", "lambda", "try"}
```

Chapter 15: Programming Project 1: Exercise15_01

Hint:

The program contains the main function and the sumDigits(n). The main function prompts the user for the input number n. It then invokes sumDigits(n) to return the sum of the digits in n.

sumDigits(n) returns n if n is a single digit. Otherwise, it returns sumDigits(n // 10) + sumDigits(n % 10). n % 10 is the last digit in n. n // 10 is the remaining number after removing the last digit.

Chapter 15: Programming Project 4: Exercise15_13

Hint:

The program contains the main function and two overloaded count functions. The main function prompts the user for the input string and then a character. It then invokes the count(const string& s, char a) function.

The count(const string& s, char ch) function invokes count(s, ch, s.length() - 1).

The count(s, ch, high) is a recursive helper function. The function returns 0 if high < 0. Otherwise, it returns count(s, ch, high - 1) + (s[high] == ch ? 1 : 0).

Chapter 15: Programming Project 5: Exercise15_19

Hint:

For simplicity, you can assume that the decimal integer is greater than 0. Your submission will work with this assumption.

The program contains the main function and the decimalToBinary function. The main function prompts the user to enter an integer then invokes the decimalToBinary(int) to return a binary string for the integer. It displays the binary string.

The decimalToBinary(value) function returns "" if value is 0, otherwise, it returns decimalToBinary(value // 2) + str(value % 2).

Note that decimalrBinary(value // 2) returns a string, value % 2 is 0 or 1.

Chapter 18: Programming Project 1: Exercise18_01

Use the code from https://liangpy.pearsoncmg.com/test/Exercise18_01.txt as a template to complete this program. You need to code the MyLinkedList class. You can check your code using the CheckExercise tool and then submit it to REVEL. When you use the CheckExercise tool, submit the entire code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.

Chapter 18: Programming Project 2: Exercise18_03

Hint: You need to submit the entire program for this exercise. Your program will likely exceed 5000 characters. To allow it to be submitted, you need to remove the comments, space lines, may even to reduce the tab space to reduce the size to 5000 characters.

Chapter 18: Programming Project 4: Exercise18_01Extra

Use the code from https://liangpy.pearsoncmg.com/test/Exercise18_01Extra.txt as a template to complete this program. You can check your code using the CheckExercise tool and then submit it to REVEL. When you use the CheckExercise tool, submit the entire code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.

Hint:

Study the code in LiveExample 18.13 in Section 18.8 using some samples, for example, 1+2, 2 * 3 - 3, etc. Modify the example EvaluateExpression.java incrementally. Once step at a time and you will know which step you are struggling.

Step 1. The operator % can be implemented similar to the * and / operators. Add the code for processing % in lines 39-47 in LiveExample 18.13.

Step 2. The operator \wedge has the highest precedence. However, note that the \wedge operator is right-associative, meaning that $2 \wedge 3 \wedge 2$ is same as $2 \wedge (3 \wedge 2)$. In lines 51-61 in LiveExample 18.13, the program processes the $*$ and $/$ operators, add the code for processing the \wedge operator after this block.

Chapter 19: Programming Project 1: Exercise19_01

Use the code from https://liangpy.pearsoncmg.com/test/Exercise19_01.txt as a template to complete this program. You need to code the MyBST class. You can check your code using the CheckExercise tool and then submit it to REVEL. When you use the CheckExercise tool, submit the entire code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.

Hint:

For the definition of the height of a binary tree, see Section 19.2. Use recursion. If the tree is empty (i.e., root is None), return -1, else return 1 + the max of the height of the left and right subtrees.

Chapter 19: Programming Project 2: Exercise19_03

Use the code from https://liangpy.pearsoncmg.com/test/Exercise19_03.txt as a template to complete this program. You need to code the MyBST class. You can check your code using the CheckExercise tool and then submit it to REVEL. When you use the CheckExercise tool, submit the entire code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.

Hint:

Copy the height() method from the preceding project. You can compare the tree size with $2^{(height+1)} - 1$ to determine if the tree is perfect.

Chapter 19: Programming Project 3: Exercise19_07

Use the code from https://liangpy.pearsoncmg.com/test/Exercise19_07.txt as a template to complete this program. You need to code the MyBST class. You can check your code using the CheckExercise tool and then submit it to REVEL. When you use the CheckExercise tool, submit the entire code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.

Hint:

Use recursion. If the tree is empty (i.e., root is None), return 0, else return 1 + the number of the non-leaf nodes in the left subtree + the number of the non-leaf nodes in the right subtree.

Chapter 19: Programming Project 4: Exercise19_11

Use the code from https://liangpy.pearsoncmg.com/test/Exercise19_11.txt as a template to complete this program. You need to code the MyBST class. You can check your code using the CheckExercise tool and then submit it to REVEL. When you use the CheckExercise tool, submit the entire code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.

Chapter 19: Programming Project 5: Exercise19_01Extra

Use the code from https://liangpy.pearsoncmg.com/test/Exercise19_01Extra.txt as a template to complete this program. You need to code the MyBST class. You can check your code using the CheckExercise tool and then submit it to REVEL. When you use the CheckExercise tool, submit the entire code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.

Chapter 20: Programming Project 1: Exercise20_07

Use the code from https://liangpy.pearsoncmg.com/test/Exercise20_07.txt as a template to complete this program. You need to code the MyBST class. You can check your code using the CheckExercise tool and then submit it to REVEL. When you use the CheckExercise tool, submit the entire code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.

Chapter 21: Programming Project 1: Exercise21_01

Hint: Your program will likely exceed 5000 characters. To allow it to be submitted, you need to remove the comments, space lines, may even to reduce the tab space to reduce the size to 5000 characters.

Chapter 22: Programming Project 1: Exercise22_01Extra

Use the code from https://liangpy.pearsoncmg.com/test/Exercise22_01Extra.txt as a template to complete this program. You need to code the main function and call the main function. You can check your code using the CheckExercise tool and then submit it to REVEL. When you use the CheckExercise tool, submit the entire code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.

Chapter 22: Programming Project 2: Exercise22_05

Use the code from https://liangpy.pearsoncmg.com/test/Exercise22_05.txt as a template to complete this program. You need to code the MyGraph class. You can check your code using the CheckExercise tool and then submit it to REVEL. When you use the CheckExercise tool, submit the entire code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.

Chapter 23: Programming Project 1: Exercise23_01Extra

Use the code from https://liangpy.pearsoncmg.com/test/Exercise23_01Extra.txt as a template to complete this program. You need to code the main function and call the main function. You can check your code using the CheckExercise tool and then submit it to REVEL. When you use the CheckExercise tool, submit the entire code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.

Chapter 23: Programming Project 1: Exercise23_02Extra

Use the code from https://liangpy.pearsoncmg.com/test/Exercise23_02Extra.txt as a template to complete this program. You need to code the main function and call the main function. You can check your code using the CheckExercise tool and then submit it to REVEL. When you use the CheckExercise tool, submit the entire code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.