

**ITSE 1359 – Lab 10 Assignment (Refer to Ch. 10 as needed):**

**NOTE:** Lab 10 requires you to work on and submit two solutions (3 files total). Read this document carefully and comply with submission and grading criteria.

Your first problem (Grades) requires you use the video provided to construct it and will result in two files. The second program (Interest Due with Exception Handling) should be solved as specified in the provided pseudocode and will result in one file.

**Program #1 - Grades:**

**(Submit 2 files: `grades_write.py` and `grades_read.py`)**

Problem Definition:

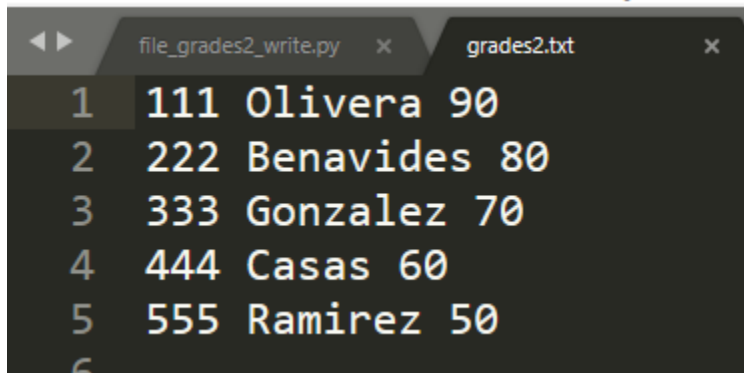
Create a two Python programs, one that writes grades to a text file and another that reads the grades from this text file and processes them as specified herein. The program that writes the grades should write them to a text file and the file that reads the text file should include an exception handling of file not found. General and specific comments required.

Description of `grades_write.py`: This is the program that will write the grades to a text file. That text file should be called `grades.txt`. This program should open the text file in write mode. It should write five records to the file consisting of student id, last name, and grade. You can use any data you like or that shown in the print screens.

Description of `grades_read.py`: This is the program that will read the grades from the text file. This program should open the text file in read mode. It should show the data in formatted report. This report should have field headers of: ID, Name, and Grade. The records should then be displayed below this followed by a count of the number of grades and the average. Columns must be aligned as shown in print screens.

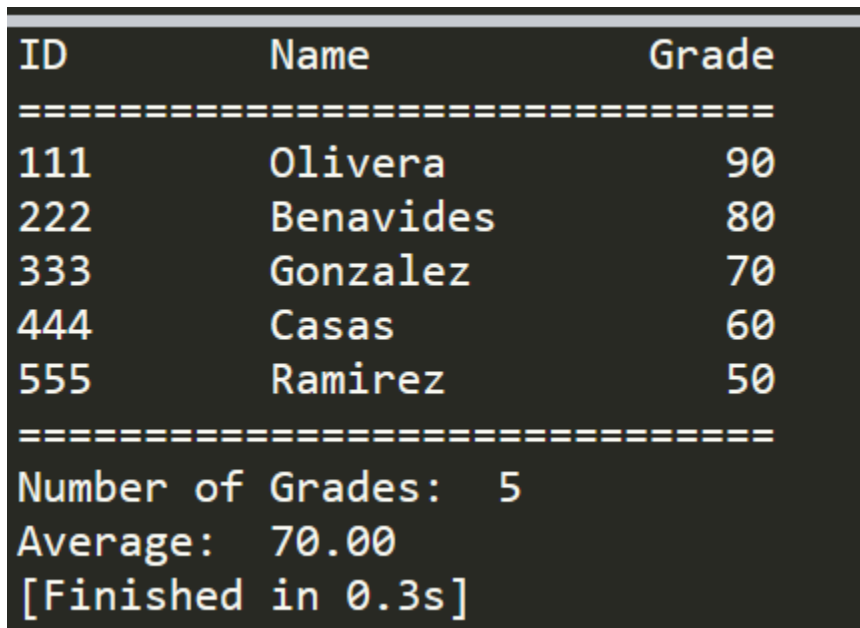
**Screenshot of Output (grades.txt):**

This is the result of running grades\_write.py:



```
1 111 Olivera 90
2 222 Benavides 80
3 333 Gonzalez 70
4 444 Casas 60
5 555 Ramirez 50
```

Note: You do not need to submit your **grades.txt** file because I will run your grades\_write.py file so that a fresh file is generated. You should code your program that creates the text file before you attempt to read it. Also, in your file that reads, its best to write your code without exception handling first and then experience errors (if file not found). Then, you add exception handling as a finishing touch to your program to make it more robust.

**Screenshot of Output (grades\_read.py)**

```
ID      Name      Grade
=====
111     Olivera    90
222     Benavides  80
333     Gonzalez   70
444     Casas      60
555     Ramirez    50
=====
Number of Grades:  5
Average:  70.00
[Finished in 0.3s]
```

**Pseudocode / Comments (grades\_write.py):**

```
# assign file name string to filename variable
# use with statement to open file in write mode and assign to file object
    # use write function to write five records
    # each record should consist of student id, name, and grade
    # each string record should have a new line escape code at end
```

**Pseudocode / Comments (grades\_read.py):**

```
# initialize counter variable for number of grades
# initialize accumulator for grades to sum the grades
# assign file name string to filename variable
# try:
    # use with statement to open file in read mode & assign to file object
    # print header line with each column formatted
    # print separator line of = characters
    # use for in loop to traverse grades
        # split record into id, name, grade
        # covert grade to int
        # print formatted detail line for id, name, grade
        # increment grades counter
        # increment accumulator for sum of grades

    # print separator line of = characters
# except FileNotFoundError:
    # print "Sorry, {filename} not found"
# generic except:
    # print "An error has occurred"
# else:
    # print number of grades
    # print average as sum of grades / grade count
```

**Code Example (grades\_write.py):**

How to open file in write mode and write a record?

```
with open(filename, mode='w') as grades:
    grades.write('111 Olivera 90\n')
```

**Code Examples (grades\_read.py):**

How do I open file in read mode and read in one record?

```
with open(filename, mode = 'r') as grades:
    print(f'{"ID":<10}{"Name":<10}{"Grade":>10}')
    print('=====')
    for record in grades:
        student_id, name, grade = record.split()
```

- The < means left align
- The > means right align
- The 10 in the f-string is the field width

How to I increment the counter and accumulator?

```
count_grades = count_grades + 1
sum_grades = sum_grades + grade
```

Is there a faster way of typing out all those "===="?

```
print('=' * 30)
```

How can I test my exception handling?

- Just delete your text file or rename your text file in the code.

```
filename = 'gradesX.txt'
```

Your test for no file should give you:

```
Sorry, the file gradesX.txt does not exist.  
[Finished in 0.2s]
```

Code that produced the above:

```
except FileNotFoundError:  
    print(f"Sorry, the file {filename} does not exist.")
```

How do I know what exceptions to handle?

- Run your code without exception handling and note the traceback. The traceback will tell you what error you got and thus what exception handler to use.

What do I submit?

- If you do experiment with the exception handling, be sure to revert your code to match pseudocode.

## Program 2 – Interest Due with Exception Handling:

**Problem Statement:** This program will start with your solution to lab 7 program – interest\_due.py. Refer to that problem statement for elaboration. Save that file as interest\_due\_except.py. In this program you will use exception handling to handle inappropriate data being entered.

**Coding:** The design of your exception handling is up to you, so long as you handle the `ValueError` exception. You may want to provide for a generic exception even though it may not ever fire. If you are going to use a simple try scheme, you should provide for the `else` clause. What you say in your feedback to exceptions caught is up to you, so long as your statements are professional.

**Output:** Your output will be the same, except that now if the user enters non-numbers, they will get feedback. Before your program would crash if you enter “ASDF”, but now the user will get a message such as “Please enter numbers only”. So, entering “ASDF” for input will result in a `ValueError` – see Traceback. Handle this, so that your program exists the crash more gracefully.

**Resources:** Refer to [chapters 7](#) for discussion of user input and [chapter 10](#) for discussion of exception handling (p. 194). You may also want to refer to the internet for syntax examples. W3schools.com offers excellent examples of Python coding structures.

**Note:** Sublime Text does not handle user input, so you will have to use an editor that can like Idle or [VS code](#) – Unless you want to run your file from a command prompt – I show you how to do that in my YouTube videos for [chapter 7](#).

**Specific Comments / Pseudocode (interest\_due\_except.py):**

# print name of program

# set repeat flag to True

# while repeat:

    prompt for inputs and convert to numeric in try block  
     catch ValueError and generic exception  
     else block computes total interest and output  
     prompt and test for again

# say thanks and goodbye!

**Example code (interest\_due\_except.py):**

```
try:
    amount = float(input("\nHow many dollars do you wish to borrow? "))
    interest_rate = float(input("What is the interest rate? "))
    years = int(input("How many years would you take the loan? "))
except ValueError:
    print("Input Error >>> Please enter numbers only. You will have to start again.")
except:
    print("An error has occurred.")
else:
    total_interest = amount * (interest_rate/100) * years
    output = f"\nIf you borrow ${amount:,.2f} at an interest rate of {interest_rate}%"
    output += f"\nfor {years} years, you will pay ${total_interest:,.2f} in interest."
    print(output)
```

**Screenshot of Output (interest\_due\_except.py):**

```
Program - Calculate Interest on Loan:

How many dollars do you wish to borrow? 2000
What is the interest rate? 3.85
How many years would you take the loan? ten
Input Error >>> Please enter numbers only. You will have to start again.

Would you like to run another calculation? (y/n) n

Thanks for using this program. Goodbye!
PS C:\Users\Presentations> █
```

**OPTIONAL Enhancement (Interest Due with Exception Handling):**

Optionally, if you are an advanced programmer you may want to place each prompt in a while loop and catch the `ValueError` in each prompt. Use the `continue` statement after catching the exception to repeat the prompt. Use the `break` statement to exit the while loop. After you do your prompts, you can perform your calculations, show output, and test for again.

```
while repeat:
    while True:
        amount = input("\nHow many dollars do you wish to borrow? ")
        try:
            amount = float(amount)
        except ValueError:
            print("Input Error >>> Please enter numbers only.")
            continue
        except:
            print("An error has occurred.")
            continue
    break
```

Output of above (see how `continue` and `break` work?):

```
Program - Calculate Interest on Loan:

How many dollars do you wish to borrow? 2000
What is the interest rate? 3 percent
Input Error >>> Please enter numbers only.
What is the interest rate? 3.85
How many years would you take the loan? ten
Input Error >>> Please enter numbers only.
How many years would you take the loan? 10

If you borrow $2,000.00 at an interest rate of 3.85%
for 10 years, you will pay $770.00 in interest.

Would you like to run another calculation? (y/n) n

Thanks for using this program. Goodbye!
PS C:\Users\Presentations>
```



## How do you prompt and test for again?

- There are many ways. You can make it simple or elaborate. It depends on how you want the program to handle the situation. Here is one simple scheme that you could use in your interest program that would appear at the end of your loop:

```
#prompt and test for again
again = input("\nWould you like to run another calculation? (y/n) ")
if again.lower() == 'n':
    repeat = False
else:
    repeat = True
```

In the above code, you can put your Goodbye statement in the body of the if that catches the “n” or as a separate line that appears as the last line of code that is not indented or part of any other coding structure so that it executes last. Choice is yours.

## Submit your lab assignment:

**VIP:** Since this lab assignment involves multiple files, create a folder called “lab10\_username” where username is your Collin username. Put your three files it (grades\_write.py, grades\_read.py, interest\_due\_except.py). Zip your folder. Using the Canvas assignment tool, upload your zipped file to the **lab 10 assignment**. After you have uploaded your zipped file, submit your lab assignment.

See last page of this document for directions on how to create a folder and zip a folder. If you do not submit right, you may get points off or no grade at all. I will not grade twice.

All lab assignments must be submitted using the CANVAS assignments tool. **Lab Assignments will not be accepted any other way.** Make sure you submit your work to the right lab assignment number otherwise you will not get credit.

## Grading Criteria:

- ✓ You must follow the Student Information video.
- ✓ Don't forget general comments.
- ✓ Specific comments are optional.
- ✓ Use white space to make your code easy to read.
- ✓ Comply with PEP-8 conventions for variable names, file names, etc.
- ✓ Satisfy the problem definition and other grading standards.
- ✓ Your work should not have syntax errors.
- ✓ Your work must be your own.
- ✓ Match your output screen to screenshot provided.
- ✓ Any deviations from lab specifications will result in points off.
- ✓ If you want to vary, do so on your own.

## VIP Videos:

Grades lab (step-by-step): [Lab 10 - Grades Video](#)

Lectures and other videos: [VIEW MY PYTHON PLAYLIST](#)

## Closing:

If you have questions about this lab send me a message using canvas inbox or attend zoom office hours. See Unit 0 for link for my office hours. Programming tutors are also available and listed in your Canvas class as announcements.

By-the-way, a great way to get ready for your lab assignments (and exams) is do the Try It Yourself problems in your book. Most of the [solutions](#) are on the authors website and I also discuss them in my YouTube video lectures.

Another way to get ready for your lab assignments is to review my lecture notes for each chapter. In Unit 0, find the link for my chapter notes.

**Warning:** Labs will increase in complexity with each lab assignment. A beginner may need 5 to 15 hours to complete each lab. Please start early because there will be no extensions.

Consider doing this lab over a three-day period:

- ✓ Day 1 – Get confused and run out of time.
- ✓ Day 2 – Research and debug errors.
- ✓ Day 3 – Polish, double-check everything, and submit.

Have an exception free day!

Prof. Benavides

## **Resources – Create a folder, put files in it, and Zipp folder:**

### **Win 10 - How do I create a folder?**

- 1.) Right-mouse click on the Desktop.
- 2.) Point to “New” and then click on “Folder”.
- 3.) Type in the name of your folder and press Enter.

### **Win 10 – How do I put files in a folder?**

- Just drag them to your folder if they are not in there to begin with.

### **Win 10 - How do I zip a folder? (Do it this way please.)**

- 1.) Right-mouse click on the folder.
- 2.) Point to “Send to” and then click “Compressed (zipped)” folder”.
- 3.) Now you see zipped file by same name as folder with a zipper on it.
- 4.) Do not send a .rar file or any other file type other than .zip.

### **macOS - How do I create a folder?**

- 1.) Right-mouse click on the Desktop.
- 2.) Select “New Folder”.
- 3.) Type in the name of your folder and press Enter.

### **macOS – How do I put files in a folder?**

- Just drag them to your folder if they are not in there to begin with.

### **macOS - How do I zip a folder? (Do it this way please.)**

- 1.) Right-mouse click on the folder.
- 2.) Select “Compress”. Name of folder appears after word Compress.
- 3.) Now you see zipped file by same name as folder with a zipper on it.
- 4.) Do not send a .rar file or any other file type other than .zip.