Computer Laboratory 01

CSCI 1913: Introduction to Algorithms,
Data Structures, and Program Development

1 Essential information

- This assignment is due Tuesday February 4th at noon and will be turned in on gradescope
- The assignment is intended to be done in pairs, although you're allowed to work on your own if you want.
- You can start the assignment before your lab period, although I recommend not getting too far – the lab TAs often have useful hints, and I don't want you wasting too much time struggling with something that the TAs might announce in-class.
- For more rules see the lab rules document on canvas.
- Files are provided to help you start and test this lab. You may struggle if you do not use these files.

2 Introduction

This laboratory is intended as a short introduction to python for students who are already familiar with a C-style language such as C, C++, or Java. The Python syntax in this lab might go beyond what's been covered in lecture formally before tuesday, but should be quite achievable none-the-less. If you find yourself getting stuck on the python syntax – remember that this is **normal**, It's OK to ask for help with the syntax or look for syntax guides to support your programming. Trust that a week or two of python programming is all it will take for the syntax to start feeling as natural as your first programming language.

You will find it useful to begin by planning out your program in pseudocode or a familiar programming language. Some of the steps in this code might be non-obvious, or require some problem solving, and it's much easier to do that before struggling with the new python syntax. Having a solid plan in this way will let you focus on ONLY the translating-into-python part of this lab. Remember, it's OK to ask the TAs for help with the python syntax, or to reference zybook or your notes for that.

Educationally, by doing this lab you should:

1. Set up (and test) your python programming environment
2. Practice the process of submitting code in this course

3. Make sure you have access to slack (so you can ask questions, attend office hours, and look at pet pictures and memes posted by the class)
4. Write your first Python functions.

Before outlining the assignment itself – you should start by setting up your python programming environment.

3 Software environment setup

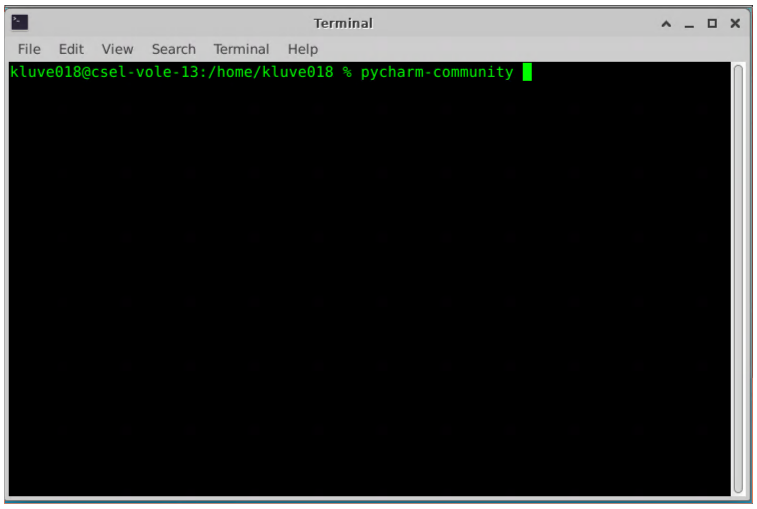
For the next few labs you will be doing python programming. Ultimately, all that matters is that you've created sufficient and correct python files. Therefore, you are free to use whichever text editor and programming environment you choose. That said, the recommended lab IDE (Integrated Development Environment) for the python part of this course is the PyCharm IDE. This section describes how to launch pycharm (on the CSELabs machines), and how to set up a new project for this lab.

3.1 Launching PyCharm

note This is where I would like to be able to confidently give you the rundown on the exact steps for launching pycharm for the first time on the CSELabs machines, but unfortunately, my (Daniel's) CSELabs account is no longer a reasonable reflection of yours (any given student's), and in past semesters we've learned that what works for me doesn't always work for students for launching programs. As such, my specific instructions may not always work. For this, it will be quite important to look to your lab staff for specific instructions, and possibly a quick demo.

It's possible that pycharm will have a good "launcher" setup in the applications menu. If so, just go to applications (top left corner), then the development submenu, and finally "PyCharm Community Edition". If that works, skip ahead to the project setup instructions.

If Pycharm doesn't seem have a launcher script set up in the application menu on CSE-Labs machines, then you will need to launch pycharm from the terminal. However, as this doesn't seem to be set up, you can just launch pycharm from the terminal with the command `pycharm-community`



```
kluve018@csel-vole-13:/home/kluve018 % pycharm-community
```

Depending on your past use of PyCharm you may need to enter some one-time settings such as color theme and default set of keyboard shortcuts. You can set these options however you want, while I prefer default key-bindings and a light theme, many people may disagree. Ultimately, these decisions have a negligible effect on your coding output, and can be changed later from the settings menu. If you are not prompted for these settings, that too is fine.

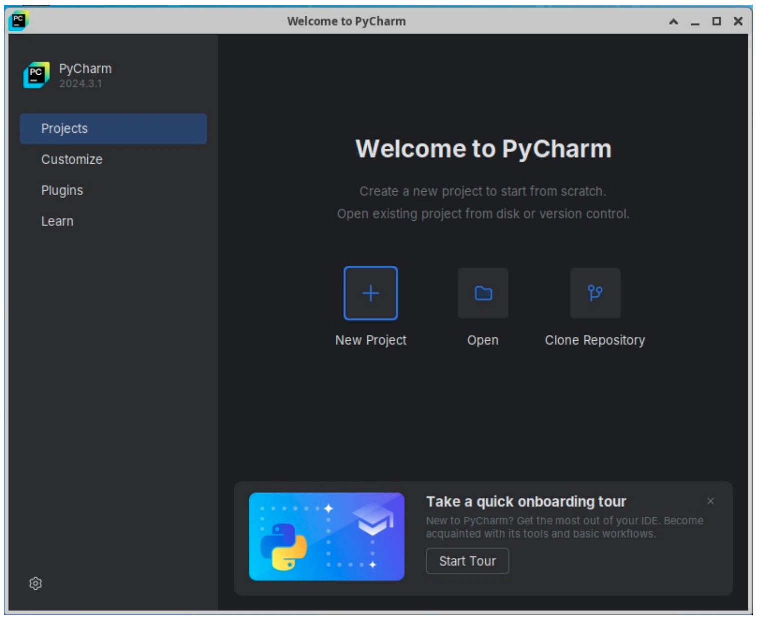
3.2 Setting up a Project

Many IDEs are organized around the concept of a "project" – a group of related code and file resources. It is rare for a programmer to work on a program that only has one file, often any interesting programming task would be split across many files, and may possibly have other associated resources (data files, configuration, etc.) These related files are, together, referred to as a project.

For this class, while labs often will be small, **I would still recommend making a different Project for each lab**. By keeping the labs separate you reduce the chance of any issues caused by reused variable, function, or file names. It is quite common for a programmer to have many projects on their computer (although it is rare for them to have more than one open at a time).

IMPORTANT As you make these projects, be very careful to know where in your file-system the files are located. This is something you will need to know later so you can turn the files in.

Once open, depending on settings and past use PyCharm will either open to it's splash-screen or a previously opened project. Either way you want to **make a new project**. If you are in a previous project go to File → new → Project. Otherwise you can simply click the "Create New Project" option in the splash screen:



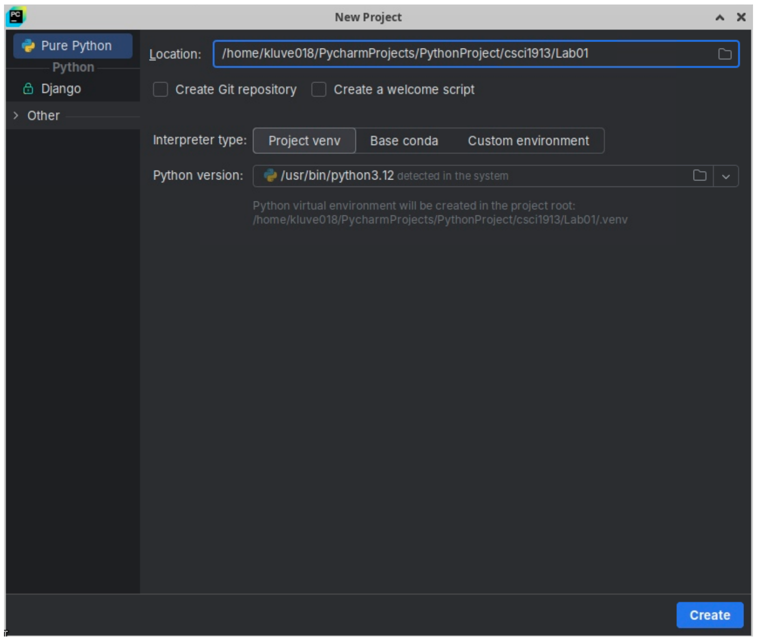
Welcome to PyCharm

Create a new project to start from scratch.
Open existing project from disk or version control.

Take a quick onboarding tour

New to PyCharm? Get the most out of your IDE. Become acquainted with its tools and basic workflows.

The new project window should look something like this:



Location: /home/kluve018/PycharmProjects/PythonProject/csci1913/Lab01

Create Git repository
Create a welcome script

Interpreter type: Project venv

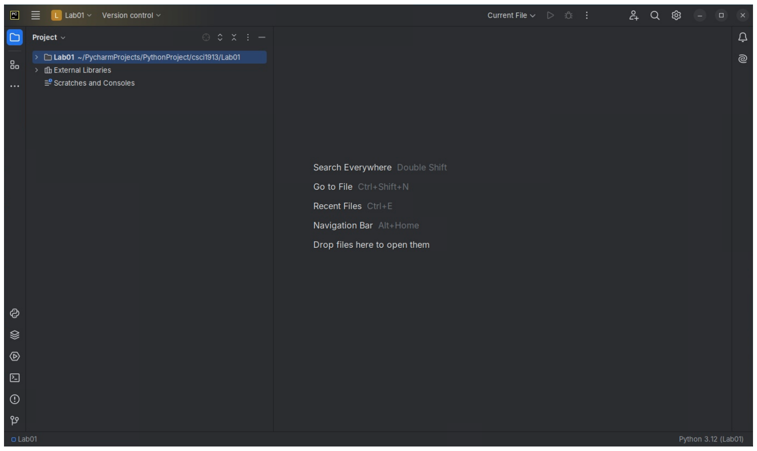
Python version: /usr/bin/python3.12 detected in the system

Python virtual environment will be created in the project root:
/home/kluve018/PycharmProjects/PythonProject/csci1913/Lab01/.venv

You can freely change the project Location, the default is fine too. Simply make sure you are confident you can find that location in your file browser. **It is very important to make a note of the project location.** You will need to find these files outside of the IDE to hand them in for grading.

You will also want to check your project interpreter by clicking the little triangle next to the project interpreter menu option. This setting effects several aspects of how PyCharm set's your project up. While most options are fine, the default option has a few extra files we don't need for our purposes. Switch your project interpreter to an existing interpreter, and make sure it's a python 3 interpreter (not a python 2 interpreter). (Note the exact python3 version shouldn't really matter, so long as it's above 3.8 or so you should be fine.)

Once you have that setting you can click next. This should bring you to the main project view itself.

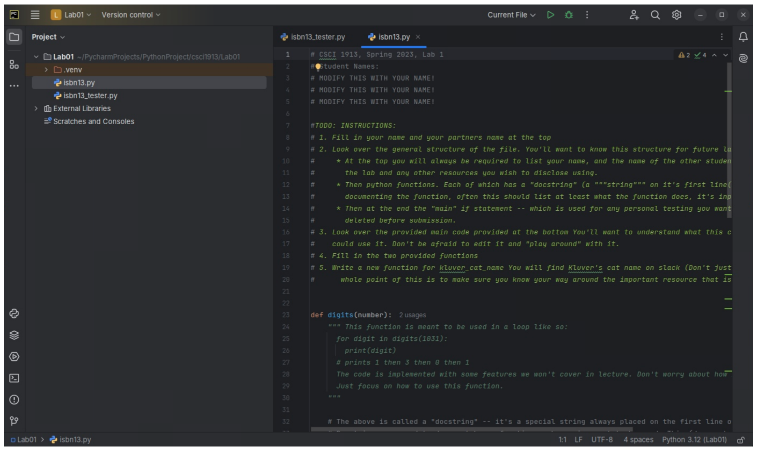


This screen has a few parts, which you will get used to as you use it more. Most importantly, however, is the sidebar on the left, which gives you an overview of project files, and the main pane, in which your files will show up. You will do at least one meaningful project in this IDE, so it will be worth spending some time getting used to as you work, instead of just trying to push past it as fast as possible.

If the side bar doesn't show up, you will want to summon it with the appropriate keyboard command. On windows and Linux, under default keybindings, the keyboard command is alt + 1. If that doesn't work you can also toggle the side bar from the menu → View → Tool Windows → Project.

On the sidebar there should be a folder icon whose label is based on the project's name (I.E. I named mine lab01, so the folder is labeled lab01, if you named yours lab1, it would be named lab1) That folder represents your main project folder in the file system. If you need to add a file to PyCharm you can simply click and drag the file from your file browser to that folder icon and PyCharm will copy that file to the right location. Likewise, to add a new file you can right-click on that icon and go to new → Python file. To view the contents of that folder you may need to click the little triangle next to the folder icon.

You will regularly need to load files info this main project folder to run. The following screenshot shows what happened after I moved the Lab01 files into place, for instance:



```
# CSCI 1913, Spring 2023, Lab 1
#Student Names:
# MODIFY THIS WITH YOUR NAME!
# MODIFY THIS WITH YOUR NAME!
# MODIFY THIS WITH YOUR NAME!

#TODO: INSTRUCTIONS:
# 1. Fill in your name and your partners name at the top
# 2. Look over the general structure of the file. You'll want to know this structure for future la
#     * At the top you will always be required to list your name, and the name of the other studen
#       the lab and any other resources you wish to disclose using.
#     * Then python functions. Each of which has a "docstring" (a """string""" on it's first line(
#       documenting the function, often this should list at least what the function does, it's inp
#     * Then at the end the "main" if statement -- which is used for any personal testing you want
#       deleted before submission.
# 3. Look over the provided main code provided at the bottom You'll want to understand what this c
#    could use it. Don't be afraid to edit it and "play around" with it.
# 4. Fill in the two provided functions
# 5. Write a new function for kluver_cat_name You will find Kluver's cat name on slack (Don't just
#      whole point of this is to make sure you know your way around the important resource that is


def digits(number):  2 usages
    """ This function is meant to be used in a loop like so:
     for digit in digits(1031):
       print(digit)
     # prints 1 then 3 then 0 then 1
     The code is implemented with some features we won't cover in lecture. Don't worry about how
     Just focus on how to use this function.
    """

    # The above is called a "docstring" -- it's a special string always placed on the first line o
```

Admittedly, it's hard to tell in this screenshot, but the lab01 files are in the lab01 folder (but not in the .venv folder, which shouldn't have lab files in it, but otherwise can be ignored for now.)

4 Python syntax notes

Leture won't have covered every piece of syntax you might need for this lab. So I've included some syntax notes here that should cover just about everything you need.

4.1 function declarations

The python syntax for declaring a functions is as follows:

```
def <name>(<parameters>):
    """ docstring -- describe the inputs and outputs """
    code
    code
    code
```

with parameters being simply a list of names separated by commas (like python variables, python parameters have no type) As an example:

```
def isDivisible(x, y):
    """A function to check if x is divisible by y, returning a
       boolean """
    return x % y == 0
```

4.2 Loops

The python for loop's basic syntax is as follows:

```
for <variable> in <something>:
    code using variable
```

In python there are *many* things you can put in `<something>`. We will see two in this lab:

```
for <variable> in digits(number):
    code using variable
```

This loop uses a function I provided (digits) which lets you loop over the digits of a number.

If you don't want to loop over the digits in a number and want a more "traditional" loop experience, you will need this version:

```
for <variable> in range(limit):
    code using variable
```

This loop will run `limit` times, with `variable` taking the values $0, 1, 2, \ldots, limit - 1$

Sometimes you want to loop from 1 up to (and including) limit. If you want to do that I recommend the following loop:

```
for <variable> in range(1,<limit>+1):
    code using variable
```

This loop will run `limit` times, with `variable` taking the values $1, 1, 2, \ldots, limit$

5 Introduction

ISBN[1] (short for "International Standard Book Number") is a standard way to number and uniquely identify books. You've certainly seen ISBN13 Numbers before on books, both in barcode format, and more usefully written out in digits above or below a barcode. The ISBN standard calls for assigning each book a 13 digit number which will uniquely identify that publication from all other books. While this might sound dubiously useful on the surface, it turns out to be massively useful when working with book data in a digital form. Having a simple single standard for identifying books, free from complexities of human language, massively simplifies many digital book management systems (or at least it's simplified every one I've ever built!).

ISBN 978-3-16-148410-0

9 783161 484100

[1]There are actually two ISBN standards, ISBN10 (the older standard) and ISBN13 (the newer standard) –we're talking about ISBN13 here

An interesting property of ISBN codes (and in-fact most barcode numbering systems) is their error detection system. While barcodes can often be scanned without error on new books, used book barcodes can sometimes be scanned incorrectly, or be so broken that a human must type in the ISBN manually. When this happens if the ISBN is entered incorrectly it can cause the book to be misidentified by the system, which can lead to any number of problems in later processing. To protect against incorrectly scanned barcodes ISBNs have a "check digit" – the final digit of the barcode.

A 13 digit ISBN is in fact a 12-digit number. The 13th digit is not *really* part of the book's unique number. Instead, the 13th digit (rightmost digit) is chosen based on the 12 other digits so that an error can be detected.

The algorithm for validating an ISBN (marking it as either valid (read correctly) or invalid (read incorrectly)) is quite simple:

1. First check that the barcode is 13 digits long, if it is not the ISBN is not valid.
2. sum up all the odd-position digits of the ISBN as sum_{odd}
3. sum up the even-position digits of the ISBN as sum_{even}
4. compute $total = sum_{odd} + 3 * sum_{even}$
5. if $total$ is divisible by 10, then the ISBN is valid.

So the barcode 9780306406157 would be processed as follows:

$sum_{odd} = 9 + 8 + 3 + 6 + 0 + 1 + 7 = 34$ (odd position digits)

$sum_{even} = 7 + 0 + 0 + 4 + 6 + 5 = 22$ (even position digits)

$total = sum_{odd} + 3 * sum_{even} = 34 + 22 * 3 = 34 + 66 = 100$

Since 100 is divisible by 10, we know that this is a valid barcode.

While this process may seem a bit arbitrary, it's mathematically well-designed. This strategy is guaranteed to notice common errors such as single-digit typos, and has a 90% chance of noticing most other forms of errors. Given that this only costs us one extra digit, and a small amount of computation, this is a remarkable amount of error detection.

6 Formal Requirements

(This is the part of this document that describes the formal requirements)

Broadly, your program will be required to have three things:

- A function for validating ISBNs (named `is_valid_isbn13(isbn)`) which is designed to be easy for any future program to use.
- A function for computing ISBN check digits (the last digit) (named `make_isbn13(number)`)
- A third function named `kluver_cat_name` which just returns a string. (This one gives you a chance to make a function from scratch, and helps make sure everyone can access Slack).

6.1 is_valid_isbn13(isbn)

This function should take a single integer (not a string) and return True or False to indicate if the number is a valid ISBN.

- The return value of this function should be a Boolean, with True indicating a valid ISBN.
- Negative numbers are not valid ISBNs
- Numbers with more than 13 digits or less than 13 digits are not valid ISBNs (There are a few good ways to check this – I recommend thinking of this as a ”numbers” problem, what's the largest 13 digit number? and what's the smallest?)
- Your function will need to apply the algorithm listed above to check if the check-digit for a number is correct.
- Your function is not expected to handle non-integer data. We will not test your functions with non-integer data, and it's OK if your function crashes or otherwise "misbehaves" when given non-integer data.

HINTS

- You will need a way to check if a number is divisible by 10. This is commonly done using the modulus operator `%` (you've almost certainly seen this trick before). You can see an example of this in the syntax help section above (we use it in the example function)
- You will need a way to loop over digits in a number – a helpful function has been provided in the lab starter file. You can see syntax for using it in the syntax guide. If you wanted to learn more about it, I might recommend starting by running the following python code and seeing what it does:

  ```
  for digit in digits(12345678):
      print(digit)
  ```

- You will need to use an extra variable to keep track of which ”position” you are on (even or odd).

6.2 make_isbn13(book_number)

This function should take a single integer (a book number) and convert it to an equivalent valid ISBN 13 (returning the new isbn13 as an integer) So for example, given the book number 978030640615 (12 digit book number) the make_isbn13 should return 9780306406157 (13 digit valid ISBN – note that the only change is adding ”7” to the end.)

- The return value and input should both be integers
- If given a number which cannot be converted into an ISNB13, this should return -1. There are a few cases where this might happen:
  - numbers which are already 13 digits or more.
  - numbers which are 11 digits or less

  - negative numbers
  - (Note – you might not need to explicitly check for these edge-cases.)
- For valid inputs, the return value should always start with the same digits as it's input, with one extra digit added to the end to make it a valid ISBN13.
- There are several ways to solve this problem **we've left this as a problem for you to solve.**

Hints

- There are a few solutions to this problem. Make sure you're planning a solution out in advance (and testing it with pencil/paper) before you start coding.
- There are only 10 possible check-digits $(0, 1, 2, \ldots, 7, 8, 9)$ so one possible solution is to try each one-at-a-time. You cna use is_valid_isbn13 to do the checking. This approach might be a bit less efficient, but tends to be easier to understand.
- It is also possible to solve this mathematically – think through what you can compute from the first 12 digits and how that relates to what the last digit must be to be correct.
- If you're struggling to see the mathematical solution, don't stress that – this isn't "math class", and there are other ways to solve the problem.

6.3 kluver_cat_name()

This function should take no input and return only a string. The string returned should contain the (case-sensitive) name of professor Kluver's cat. You should be able to find this information on slack without too much trouble. If you can't find it – ask the TAs for **help finding this information** (the TAs will not just tell you what my cat is named)

I'm aware this requirement might sound petty, vain, and a bit random, but there's actually two reasons behind it. The first purpose of this function is to make sure everyone has access to slack, and to get everyone to poke around slack, just a little bit. While there are rarely any access issues with slack I wanted to get those issues addressed as soon as possible (as slack is REQUIRED for online office hours, and is a key Q&A resource in this course). By putting required information from this assignment on slack we can make sure everyone checks their access to this important resource immediately, and that we are pro-active about resolving issues. I've seen student wait to access slack in the past, and they've uniformly told me "I wish I knew I could ask questions here sooner!" If you have trouble accessing slack – check the "sign up" link provided in both the syllabus and the course technology document on canvas, or reach out for assistance over email.

You will be in charge of writing the entire function declaration for this function. (This is actually the second reason for this function – practice the function syntax with a trivial function-body) Remember – this also includes a docstring – your code will run without the docstring, but it's considered bad code style and bad manners. More importantly, one of the tests formally checks if you have a docstring, and missing this will ALSO lose you points in the code style part of manual evaluation.

7 Grading

Grading on this assignment will be 50% automated tests, and 50% manual checks. The automatic tests are effectively the same as the tests seen in the tester file. The point allocation is broken down as follows:

- 8 points for turning the file in, having the right file name, the right function names, and a file that loads.
- 24 points for is_valid_isbn13
- 10 points for make_isbn
- 8 kluver_cat_name

The Manual grading will be 10% code style, and 40% spot-checks / partial credit. The spot-checking will focus on

- Bad approaches to the problem. This can take the form of correct solutions, but vastly over-complicated/inefficient, or cases where you coded to pass the tests, but failed to actually solve the problem as stated in the PDF.
- Things that make the code *anything less than a joy to read* – bad variable names, problematic code duplication (especially where this could have been avoided by calling a function). poor use of "formatting" features like newlines, whitespace, etc.
- Partial credit for code that fails tests, depending on severity of test-failure.

For code style, we will use the following style guidelines for this lab. Future labs may have a **more restrictive style guideline** (as will most workplaces).

- Every function should have a docstring (if you are not confident you know what this is, just google it!)
- Every file should have a comment at the top, which should say, at a minimum, your name and your partners name.
- Variable names should be as descriptive as possible (clearly `n` and `i` will be accepted here as common loop variables or math variables, but `c` instead of `count` would be considered a bad variable name choice)
- Reasonable use of whitespace – don't spread the lines of a function out with massive whitespace, don't use 1-space-only for indentation. That sort of thing.
- If there are problems that actively make your code hard to read, we reserve the right to mention them, even if they are not listed here. Ultimately, code is about communication, if your code doesn't make sense it's not correct (no matter how well it runs!)

8 Submission

Before submitting your work:

- Please use the tester code on canvas, and any tests you devise on your own, to test the code and carefully check that your code is bug-free. Buggy code is worth very little in the programming community – especially if the bugs prevent it from running.
- Make sure the file is named `isbn13.py` (case sensitive)
- Make sure the that you name, and your partner's name, are in a comment at the top of the file
- If you and your partner are submitting one copy of the assignment – make sure whoever submits it marks their partner **in gradescope itself**. Please do this unless you and your partner are turning in different files.

You are only required to submit the `isbn13.py` file, which will be submitted through gradescope. You can submit as many times as you want, only the final submission will be considered for grading.

If you worked with a lab partner, gradescope has a way to add your partner to your submission after you submit (to show yourself as having worked in a group). We prefer you use this feature if at all possible. Not using it may lead to trouble getting credit for your work. More information on this feature can be found here: `https://help.gradescope.com/article/m5qz2xsnjy-student-add-group-members` If you worked with a lab partner, please make sure both you and your partner have a copy of the lab files for future reference.

If you finish this lab during the lab time, feel free to notify your lab TAs, and then leave early. If you do not finish this lab during the lab time you are responsible for finishing and submitting this assignment before the deadline.