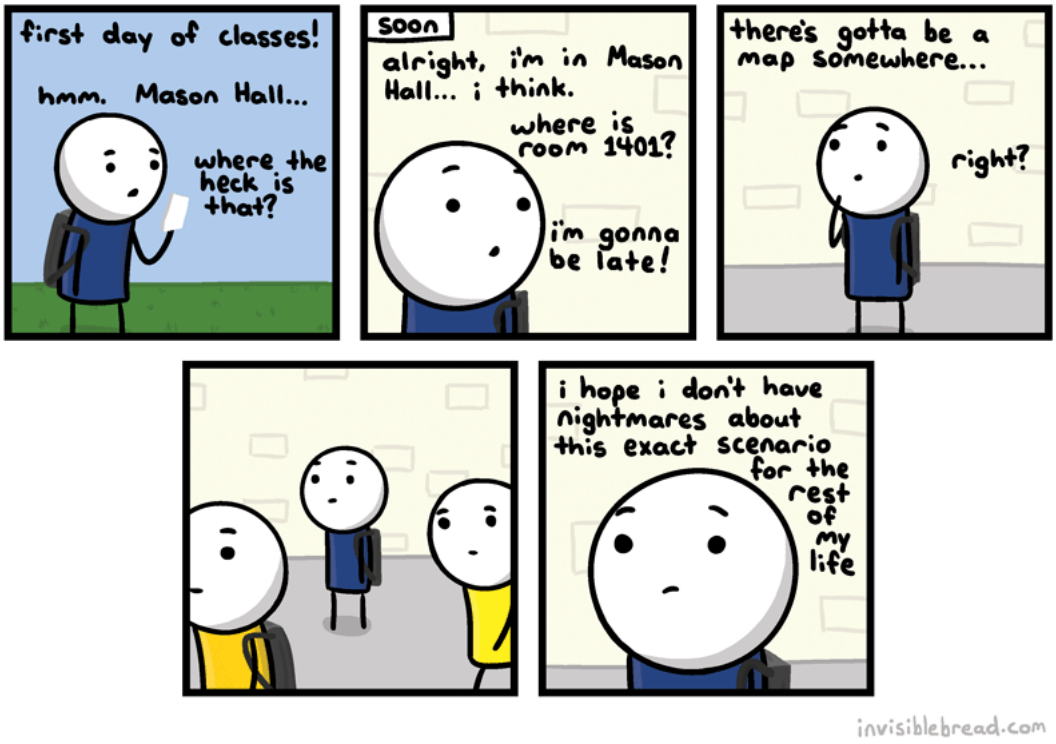
CS50 - Assignment 1
Getting started

Due: September 7th at 12pm (noon)



http://joyreactor.com/post/954711

For our first assignment, we will walk through the basics of Python and VS Code, play with Python a bit in this environment, and then make sure everything is set up to submit using Gradescope.

Assignment overview

For this first assignment we'll be doing a few different things:

1. Install Python and VS Code. *(Group assignment)*
2. Play a bit with the Python shell. *(Group assignment)*

3. Create a VS Code project and a virtual environment. *(Group assignment)*
4. Practice your understanding of expressions and their types. *(Individual assignment)*
5. Write our first program. *(Individual assignment)*
6. Get some practice with closely reading Python code. *(Individual assignment)*
7. AI and coding *(Individual assignment)*
8. Familiarize yourself with Gradescope for submitting assignments. *(Individual assignment)*

   For the group work, there is nothing to submit, you just need to show up to your learning community session and participate. For the individual work, you will submit all of your examples in a file call `assign1.py`.

1 Install Python and VS Code (*group*)

Most of the assignments in this class will use Python and VS Code.

- *If you are planning to use your own computer for this class*, you'll need to install Python and VS Code (see the instructions below). Don't wait until the last minute to do this in case you run into any problems.
- *If your are planning on using one of the computers in the CS labs* (Edmunds upstairs, 227 and 229), then you can skip the installation steps. You should be able to log into these computers with your normal Pomona login. Don't wait until the last minute to try and log in. If you do run into any issues, reach out to Corey LeBlanc in Edmunds 218 during regular business hours or the ITS helpdesk for other hours.
- *If you are also in CS51 this semester*, the instructions are more or less the same for installing Python and VS Code, so you only need to do it once.

1.1 Installing Python and VS Code on your own computer

You'll need to first install Python and then VS Code.

1.1.1 Installing Python

The latest Python interpreter can be downloaded from `https://www.python.org/downloads`. Once you visit this link, your operating system will be automatically detected. You should select the latest distribution of Python 3, Python 3.14.x, and install it after it is downloaded.

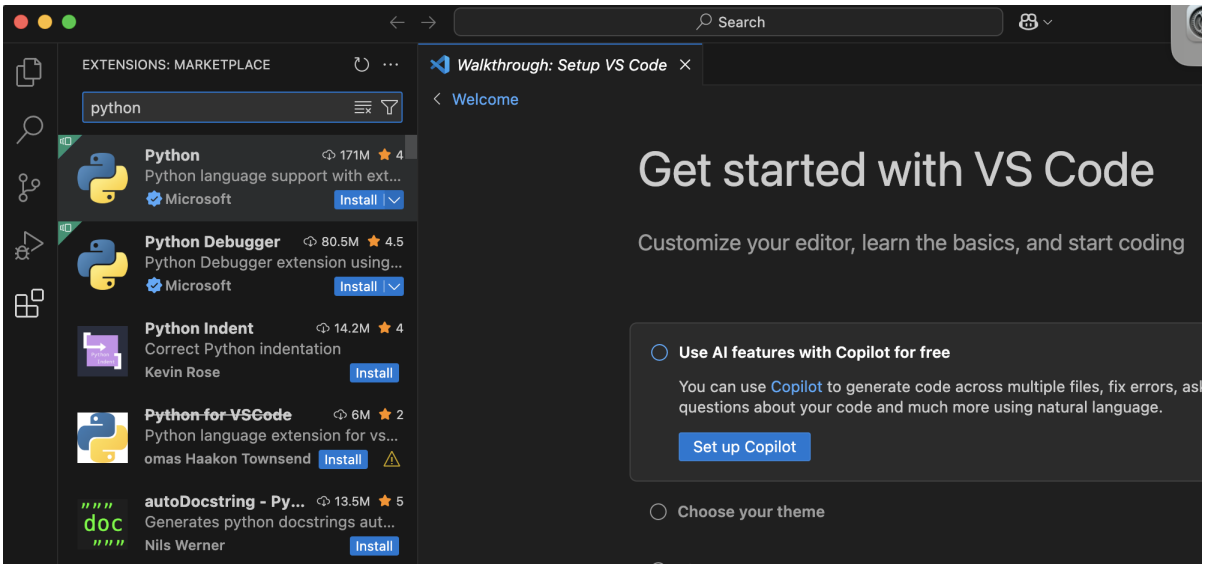
For Windows users, be sure to check the *Add to PATH* checkbox when installing Python!

1.2 Installing VS Code

Visit `https://code.visualstudio.com/download`. Your operating system won't be automatically detected. Install VS Code and then open it.

When you first run VS Code, you will be prompted with a number of options. *Make sure you do* ***NOT*** *set up Copilot* (i.e., do not set up "Use AI features with Copilot"). If VS Code asks you to log in, just skip it.

VS Code is a general-purpose IDE that can be used for a number of different programming languages. To use it for Python, you'll need to install the Python extension. Visit the extensions tab shown in the leftmost column and type `Python`. You want to install the Python extension package by Microsoft, as seen below.



Once this step is complete, you can confirm that Python (along with some other useful extensions that will come in handy later in the course) has been installed, and you can now write code in Python using VS Code.

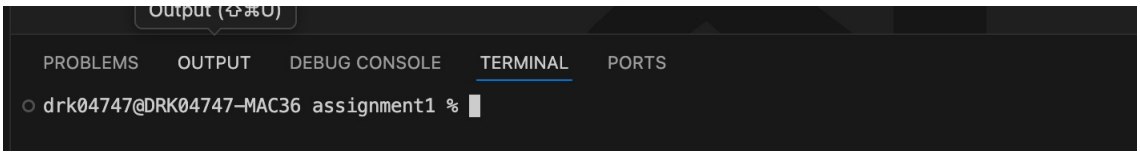
2 Play a bit with the Python Shell (*group*)

The most common way to interact with Python is to write code in a file and then run the file through Python. This is what we will do for a majority of the assignments in the class. However, Python can also be used interactively via the Python shell. When using the shell, you can type commands one at a time and the shell will "echo" the value that is returned from the command (if there is one). Playing with the shell is a great way to test out simple commands and we'll also use it to interactively test out our programs as we write them.

2.1 Terminal

The terminal is a program that provides a command-line interface (CLI): it uses text-based commands rather than a graphical user interface (GUI) to interact with the operating system. The terminal is extremely powerful and allows you to do everything that the GUI does, plus a lot more, despite a steep learning curve at the beginning. For this class, we'll mostly just use it to launch Python interactively.

To use the terminal in VS Code, go to `View` and then `Terminal` or use the shortcut `Ctrl+‘` for Windows or `Control+‘` for macOS.



2.1.1 The Python shell

In your terminal, type `python3` (if that doesn't work with some error like "command not found"), try `python`). If you are on Windows and are facing issues in this step, jump to the Troubleshooting subsection and finish the steps there before coming back.

- Try a few simple mathematical equations (e.g. "1+1", "2**3", "(100/20)+45*7"). Notice that Python makes for a pretty easy to use calculator.
- 22/7 is an approximation for π. Type this in.
- Remember that we can use variables to store intermediary values. Assign 22/7 to a variable called `pi`. Use that variable to calculate the area of a circle with radius 15 (remember, the area of a circle is π times the radius squared).
- In 2011, Pomona built a new parking structure. For a variety of reasons, the college decided to put a field on top of the parking structure (pretty cool, right?!).

Since we're in southern California, we don't need any covering for this field. However, let's pretend that we anticipate bad weather more regularly and we decide we need to cover it with a dome (global warming, maybe?). If we make the simplifying assumption that the field is a circle of radius 100 feet, what is the surface area of the dome that would cover it?

Hint: the surface area of a *sphere* is $4\pi r^2$.

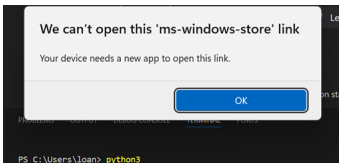
In the Python shell, you can use the up and down arrow keys to revisit commands you typed previously. Use the up arrow key to get your previous statement and then edit it to get the surface area of the dome assuming that the radius is 200 feet.

When you want to exit the Python shell, type `exit()`. If you have done something exceptionally time-consuming, e.g., you tried to calculate (`10**10**1000`) (don't!), chances are the Python shell or even your entire computer will slow down and might even crash. You can always hit `Ctrl/Command + C` to try to kill the process, but it's better to avoid such risky moves altogether.

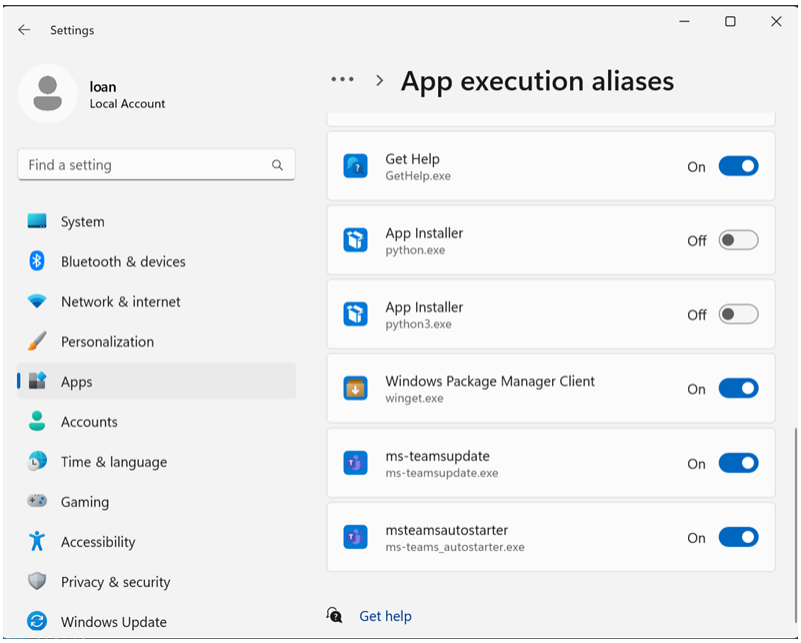
Note that there is nothing to explicitly submit for this section.

2.2 Troubleshooting working with the Python shell in Windows

If you are on Windows, there is a chance that you might get a pop-up window saying `We can't open this 'ms-window-store' link` when typing `python3`.

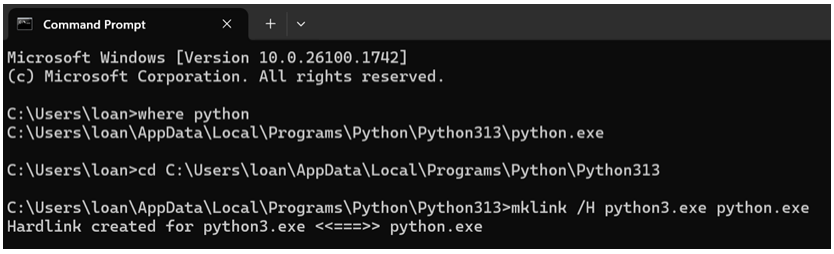


If that's the case, go to your search bar and type `App execution aliases`. You should uncheck the options `App Installer` for `python.exe` and `python3.exe`



You might still not be able to type `python3` and have it recognized. Go to your search bar and type `cmd`. The `Command Prompt` should launch.

Start by typing `where python`. This command will tell you where Python was installed. For example, for me it is in the directory `C:\Users\loan\AppData\Local\Programs\Python\Python313` but it might be a different directory for you. Copy the result you got (excluding the `python.exe`) and type `cd` and your Python path. Finally, type `mklink /H python3.exe python.exe`.



Close VS Code and relaunch it. You should be able to type `python3` now and have it recognized. If not, please reach out to your LC leader or any other member of the course staff.

3 Create a VS Code project and a virtual environment (*group*)

Working with the Python shell is great if we want to use Python as a calculator and for some superficial work, but our needs as programmers quickly go beyond that. We will now see how to create a Python project and write and run code in files.

3.1 Creating a folder for your work

Python files are just normal text files with a `.py` extension that will reside on your computer like any other files. We'll create a folder for this class where you can store all of your assignments in one place and then we'll point VS Code at that.

- In your `Documents` folder, create a subfolder called `cs50`.
- Copy or move the folder that contains this PDF into your new `cs50` folder. (You can download it again from: `https://cs.pomona.edu/classes/cs50/assignment1.zip`

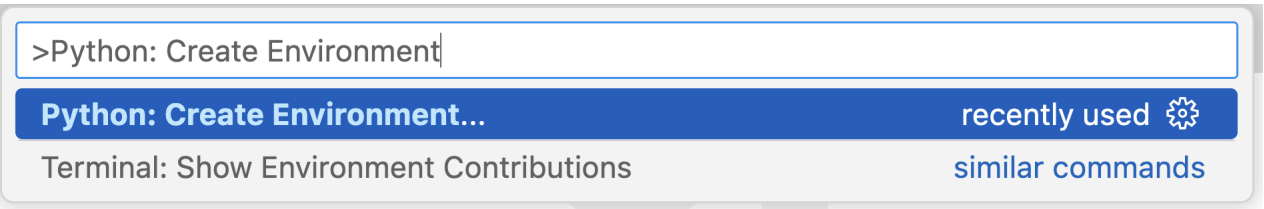
Each time you have an assignment, you'll make a new folder in your `cs50` folder for the work for that week (this will typically be the downloaded starter code for the assignment).

3.2 Creating a virtual environment

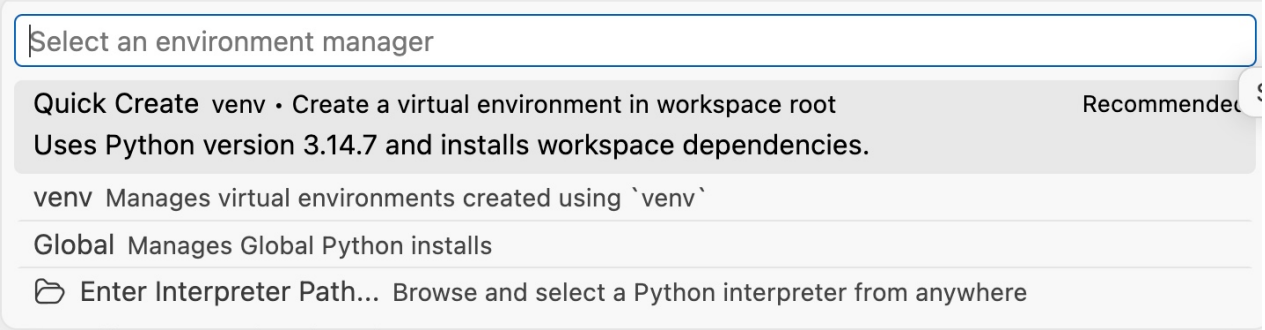
As we write more advanced Python code, we will find ourselves needing to install packages. To isolate changes to the specific projects we are working on, it is a good idea to create a *virtual environment*.

In VS Code, go to `File>Open Folder...` and select the newly created `assignment1` folder under `Documents` and `cs50`. If prompted, agree that you trust the authors in this folder. The appearance of the program may change some; this is to start you off with some customizations that make VS Code suitable for use in CS50.

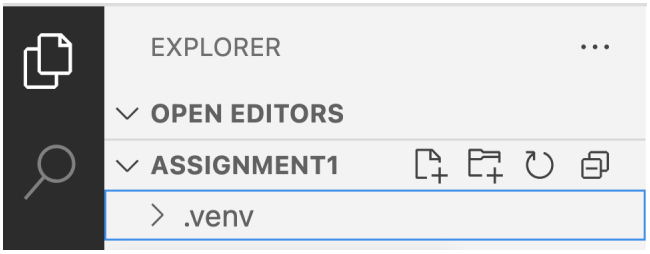
Bring up the `Command Palette` (on macOS: Command+Shift+P, on Windows: Ctrl+Shift+P) and type `Python: Create Environment`.



Then, select "Quick Create venv..." (or, if the Command Palette offers a choice between `venv` and `conda`, hit enter (return) to choose `venv`. Press enter again if it asks which version of Python to use).

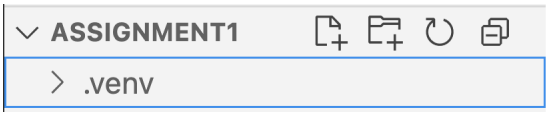


You can confirm that the virtual environment has been selected by looking in the Explorer (you'll see the `.venv` below the assignment folder):



3.3 Creating a new Python file

From the File Explorer toolbar, select the left-most `New File` button on the assignment1 folder (it looks like a page with a plus sign):



Name your file `assign1.py` (make sure you name it exactly this). The `.py` extension indicates that this file is not just plain text, but source code written in Python. It helps IDEs like VS Code highlight and make our Python code more readable.

Once you create the file, you will get a new empty file. Add a print statement to the top of the file:

```
print("This is my first Python program!!")
```

Select `Run Python File` from the top right:

In the Terminal window, you should see your text printed out.

Instead of using the VS Code `Run` button, you could have also typed on the terminal
`python3 assign1.py`
which is what VS Code runs behind the scenes.

If you wanted to run your code, but then be able to still interact with the Python sell, you could have also run it interactively using
`python3 -i assign1.py`
We'll see this interactive version can be helpful for debugging.

As an aside, you may have noticed the little gear icon next to the selected command in the command palette. You can click that icon to set a convenient key binding for the command (for example, you might choose the function key `F5` or a key combination like `Ctrl-R`).

Once you're comfortable running programs in VS Code, delete any print statements that you've added. They should not be submitted in your final submission.

4 Types in Python *(individual)*

From this point on, do *not* work with your learning community on the code and problems ahead of you. Repeated for emphasis: do not discuss code from here until the end of assignment 1 with your classmates! If we see substantial duplication of code across students, it will be considered as an academic honesty violation!

Everything in Python has a type. Run the Python shell again in the Terminal so that you can play with Python interactively. For each of the expressions below, see if you can figure out what the value will be (i.e., the result) as well as the type of that value will be. Then, check your answers using the shell. Remember, the `type` function will tell you the type of a value, e.g., you can type `type(3/2)` to figure out the type of the result of the expression.

1. `3/2`
2. `3//2`
3. `3.0/2`
4. `3.0//2`
5. `"3/2"`
6. `1.0 + 2`
7. `3.0 * 4 + 5`
8. `3 * (4 + 5)`
9. `3 / 2 + 1`
10. `"My favorite number is:  " + str(2**2**2 * 4 - 17)`

Put your answers as *comments* in your `assign1.py` with a comment above them stating that they are your type answers. For example, the beginning if your comments might look like:

```
# Type answers
# 1. 1 of type int
# 2. ...
```

5 Dr. Seuss in code *(individual)*

The following is the introduction to the book "Green Eggs and Ham" by Dr. Seuss:

```
I am Sam. I am Sam. Sam-I-Am.
That Sam-I-Am! That Sam-I-Am! I do not like that Sam-I-Am!
Do you like green eggs and ham?
I do not like them, Sam-I-Am.
I do not like green eggs and ham.
```

For the rest of the assignment, your goal is to write a program that prints out this introduction so that your program has the lowest "score". The score for your program will be the sum of the number of variables you use plus the total length (in characters, including spaces, punctuation, etc.) of all of the strings in your program. When "run" (by clicking the arrow in VS Code), your program should print out the above lyrics with the same exact formatting and capitalization.

Restrictions: Expressions used to define variables may not have any spaces in them *unless* they combine exactly two other variables with a space between them. For example, if you've defined

```
sam = "Sam"
am = "am"
```

then

```
am_sam = am + " " + sam
```

would be legal, but the following would **not** be legal:

```
am_sam_bad = "am Sam"
am_sam_bad2 = "am " + sam
```

The easiest way to recreate the first line would be:

```
print("I am Sam. I am Sam. Sam-I-Am.")
```

which would receive a score of 29, since it uses 0 variables and the string contains 29 characters.

Another option to recreate the first line would be:

```
sam = "Sam"

print("I am " + sam + ". I am " + sam + ". " + sam + "-I-Am.")
```

which would receive a score of 21. It uses 1 variable, plus strings totaling: $5 + 7 + 2 + 6 = 20$. Of course, it's possible to do much better!

You will be graded based on:

- Correctness: whether your program correctly prints out the text
- Score: the lower the better
- Creativity/effort of your solution

There are *many* possible ways to solve this with varying scores. We *do not expect* you to get the optimal lowest score, but you should make some attempt to minimize the score. Make sure you use good variable names to help keep track of what is stored in them.

When you're happy with your solution, put the final score you think you achieved in a comment at the bottom of your program.

6 AI and coding *(individual)*

AI is a powerful tool that can assist in a range of activities, including coding. For most of the assignments in the class, you could ask AI to solve them and they would do a pretty reasonable job. The reason for the assignments is not simply to generate the solutions, but for *you to learn*; having AI tools help you detracts from that learning.

The AI policy for this class strictly prohibits using AI on the assignments (including googling answers, which also often triggers an AI response). However, for this one problem, you may use AI (in fact, you need to).

Solve the following question *using AI* and add the solution to your assign1.py at the bottom.

Write a function called called *catalan* that prints out the first 100 Catalan numbers.

We want you to reflect on what you got out of this process. Add your answers (in comments below your function) to the following questions:

1. How long did it take you to solve this problem?
2. What did you learn about Python and coding in solving this problem using AI?

3. Could you answer the question "What is a Catalan number?" without looking it up?

4. Are you confident that your solution is correct? Why or why not?

Note: If you are uncomfortable using AI for whatever reason, you can instead briefly explain in 2-3 sentences your main concerns about AI.

7 Reading Code *(individual)*

Code is read many, many more times than it is written; even in this assignment, you probably read over your Dr. Seuss solution repeatedly while modifying it, rather than writing it all in one go. In a certain famous computer science textbook ("The Structure and Interpretation of Computer Programs"), the authors argue that "Programs must be written for people to read, and only incidentally for machines to execute." In the computing industry and in open-source software development, *code review* is a much more important, time-consuming, and difficult task than merely writing code in the first place. Whether you are writing code to be understood by a professor, a colleague, or even a future version of yourself, there will always be some audience.

It will take us some time before we can develop good *style* in writing our programs, but in the meantime it would be extremely helpful to practice reading and understanding code.

The final part of this assignment will give you several examples of Python code that has subtle problems. In each one, identify the issue and write a correct version of it. There is no particular need to *execute* these functions in the Python interpreter; it is enough simply to show some thought on each problem (and, of course, to type them out). *Every code example given has at least one problem that will cause a runtime error or wrong output, and that problem will be in the function's body, not its name or argument list.*

```
# Problem 1. This function should take a string and a number
# as arguments and return a greeting message.
def hello(name, age):
  return "Hi, I'm "+name+" and I'm "+age+" years old."

# Problem 2. This function takes three strings as input, and
# should return a string that combines the first two with the
# third. E.g., combine("a", "b", " ") should return "a b".
def combine(s1, s2, sep):
  result = result + s2
  result = s1 + " "
  return result

# Problem 3. This function takes two numbers as input and
# adds them together.
def addup(n1, n2):
  return str(n1) + str(n2)
```

```
# Problem 4. This function uses the previous three functions
# to deliver a complete greeting.  If all three of the above
# functions were correct, there would still be a problem
# with this one.  Read it very closely!
def greet(first_name, last_name, half_age):
  age = addup(half_age, half_age)
  name = combine(first_name, last_name, " ")
  hello(name, age)

# Problem 5. Write a fixed version which is as
# short as you can possibly write it.
# (Hint: Imagine calling this function with a specific value
#  like "hello" or 4.)
def triple(a):
  a = a + a
  a = a + a
  return a
```

Comments

Commenting is an important of coding. We'll talk more about these and give examples as we get further into the class. For this first assignment, you should have:

- Comments at the very top of the file stating your name and the assignment number.
- Your commented solutions to the type question.
- A comment delimiting (visually separating) the Dr. Seuss solution from the type answers.
- A comment delimiting the catalan solution and responses from the Dr. Seuss solution.
- A comment delimiting the fixed functions from "problems" and the catalan solution.

8 When you're done

For this assignment, you will be submitting one file: `assign1.py`. We will be submitting all assignments through https://www.gradescope.com/; the specific link for the course will be available on your section's course website. Grading and feedback will be returned through Gradescope as well. In addition, you will be able to submit regrade requests through Gradescope. You should have received an email notification that you were added to the Gradescope site with directions on how to login. Please let us know ASAP if that is not the case.

When you log into Gradescope, you should see CS50 on your dashboard. If you click on the course, you'll see the list of assignments that you have submitted and work that is due in the near term (for now, just Assignment 1).

To submit the assignment, click on the assignment name and a window will pop up for you to submit the code, via "Drag and drop" or "Click to browse", as shown in the following image.

Some assignments (in the future) may require you to upload multiple files. In this case, you should submit all the required files together at once, rather than submitting them one by one. With "Drag and drop", you can select multiple files to perform the operation. With "Click to browse", you can select multiple files by pressing "Command" (on Mac) or "Ctrl" (on Windows) and clicking on multiple files to select them for uploading.

Remember, your assignment file should be in `Documents/cs50/assignment1/`.

Note that you can resubmit your work many times as long as it is before the deadline. The very last submission will be considered for grading.

Note: Most assignments (including this one) will run an autograder after submission and give you *some* feedback on whether your submission is correct. After submitting, check the results to make sure that everything submitted correctly and the autograder tests ran appropriately. If anything fails, figure out what the issue is and resubmit. If everything passes, it mean you've passed the basic tests, though it doesn't mean that everything is perfect. One of the goals for this class will be helping you convince yourself that your solutions are correct.

9 Grading

	points
4. Types in python	5
5. Dr. Seuss in code	5
6. AI and coding	5
7. Reading code	5
Correct submission	1
Comments	2
Total	23