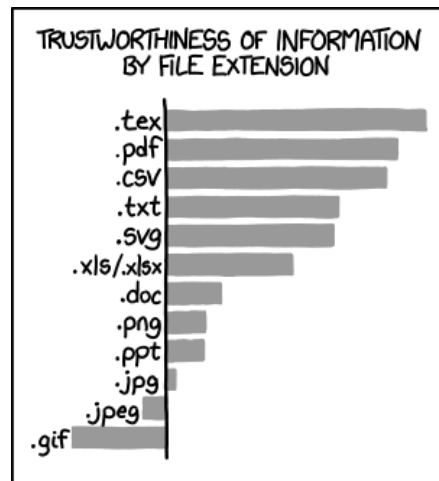


# CS51 - Assignment 1

## Getting started

Due: September, 4th at 11:59pm



<https://xkcd.com/1301/>

For this assignment, we will ensure that you:

- can log in to the lab machines
- have a working Visual Studio (VS) Code integrated development environment (optionally on your own personal machine)
- can use the command line and know basic commands
- can use the Python shell
- can create a VS Code project and a virtual environment
- can create and execute simple Python programs
- have a fundamental understanding of LaTeX
- can use Overleaf to write a simple document that will give us a glimpse of who you are
- can properly use Gradescope, the automated submission system

## 1 Introduction

This course will have weekly assignments. Lab time each week will be used to start the assignment, and you are expected to stay for its duration to receive lab credit. The work needed to complete an assignment is likely to take longer than the allocated lab time, in which case, you will finish and submit it later. If you have questions, use the lab time, instructor office hours, mentor sessions, and Slack.

In general, for lab time to be as effective as possible, you should review the lecture notes from the past week and the assignment before coming to lab. If you prefer, you can work on your personal machine. You will also benefit from having access to a pen and paper or a whiteboard.

We ask you to keep track of how long you have worked on the assignment and submit this information along with the deliverables of this assignment.

## 2 Logging in

You should be able to log into any lab machine using your Pomona account. It does not matter which lab machine you log in to. Any lab machine should recognize your account and take you to the same Desktop.

The first time you log in to a lab machine, the system may be very slow, as it creates a new Desktop for you (and because many other students may be doing this at the same time). If you attempt to log in to a lab machine before the lab, you can (a) get this over with and (b) deal with any problems before the actual start of the lab. If you are unable to log in to a lab machine with your Pomona account Username and Password, you need to contact Corey LeBlanc in Edmunds 218, during regular business hours, or the ITS helpdesk.

In this class, when writing code in Python, we'll work within Visual Studio (VS) Code. VS Code is an integrated development environment (IDE), that is, a text editor that you can use to write, run, and debug code while having nice features like syntax highlighting, code completion, and version control that will make you a more effective and efficient programmer. VS Code uses extensions to support coding in different languages, not just Python; chances are that if you continue learning more programming languages, it will remain the IDE of your choice.

Python and VS Code are already installed on the lab machines. If you choose to use the lab machines, then you can skip the next section and start working on the lab. However, we highly recommend that you install Python and VS Code on your personal machine, as it is more convenient and easier for you to access your work without needing to physically be in Edmunds.

## 3 Setting up your personal machine

If you want to use your personal machine when writing code in Python, you will need to download and install the latest stable version of Python 3 and VS Code.

Don't hesitate to ask for help from the course staff if you are stuck at this or any other step.

The latest Python interpreter can be downloaded from the official Python website. Once you visit this link, your operating system will be automatically detected. You should select the latest distribution of Python 3, Python 3.13.x, and install it after it is downloaded.

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

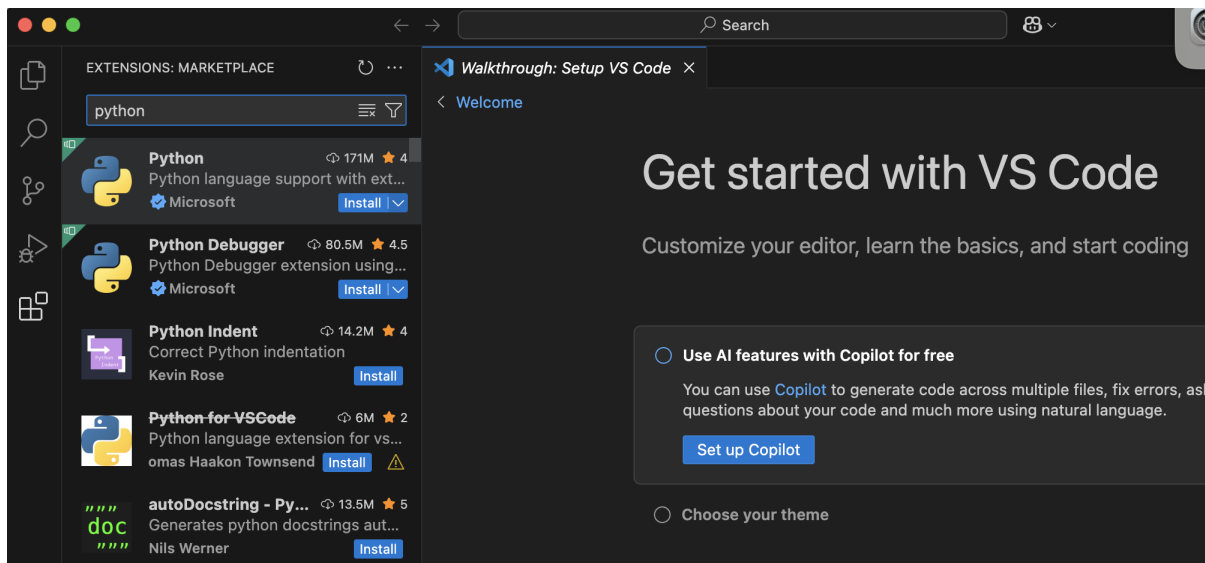


Next, visit the official Visual Studio Code website. Your operating system won't be automatically detected. Instead, you want to choose the right version for Windows, Mac, or the different distributions of Linux, download it, and install it. Ask us if you are unsure.

Once you launch VS Code, you will be prompted with a number of options. We bring your attention to these two:

- **Use AI features with Copilot:** You should **not** set up Copilot.
- **Choose your theme:** Feel free to customize, but be mindful of the legibility of your screen for the course staff and classmates during partner assignments.

Finally, you want to 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.



## 4 The command line

Either on a lab or your personal machine, go ahead and launch VS Code if you haven't already done so. Some tips: choose **Keep in Dock** for macOS and **Pin to Taskbar** for Windows so that it's always accessible. And under **File**, choose the option **Auto Save**.

Next, we will open the *terminal*. The terminal is a program that provides a command-line interface (CLI), that is 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.

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

We will learn four basic terminal commands.

- **pwd**: short for print working directory. It prints your current location (i.e., the folder you are currently in).
- **ls**: short for list. It lists all the files at your current location.
- **cd**: short for change directory. It moves you around your computer from folder to folder.
- **mkdir**: short for make directory. It creates a folder in the location you are in.

The *prompt* is the bit of text on the left that's waiting for you to type something into the command line. For example, in my personal computer, that would look like:

```
apaa2017@APAA2017-MAC21 ~ % █
```

## 4.1 pwd command

As we said, the **pwd** command is short for print working directory. It prints your current location (folder). Try it:

```
apaa2017@APAA2017-MAC21 ~ % pwd
/Users/apaa2017
_
```

You'll see the location at which your terminal and command line are currently running. For example, I am in the *home directory* **apaa2017** (your home directory name will likely be something very different and related to your username).

Note that I am using macOS here. The formatting will be different if you are on Windows or Linux. The slash characters (/) show subfolders. On Windows, they're usually backslashes (\) or double backslashes (\\). For example, on Windows, my home directory looks like **C:\Users\apaa2017**. Thus, I'm currently in the subfolder named **apaa2017**, within the folder named **Users**, on the drive named **C:** (Windows puts colons after the names of disk drives).

Notice that after you type **pwd** and get the results, the next prompt appears, waiting for another command...

## 4.2 ls command

The **ls** command stands for list (all files in the current directory). On my personal computer, I get a gazillion directories/folders (we use these terms interchangeably) and files listed:

```

apaa2017@APAA2017-MAC21 ~ % ls
_output
alexandra_papoutsaki@alumni.brown.edu - Google Drive
alexandra.papoutsaki@pomona.edu Creative Cloud Files
alexpapster@gmail.com - Google Drive
anaconda3
AnacondaProjects
Applications
Box-Box-Backup
Calibre Library
Creative Cloud Files Company Account AICCU - Pomona College alexand
DF@pomona.edu
Desktop
Documents
Downloads
Dropbox
eclipse
eclipse-workspace
git
go
Library

```

Things will look slightly different on Windows. Many of these directories are created by default by your OS, but as we will soon see, we can create our own directories through the terminal and see them listed. Go on and give it a try.

### 4.3 cd command

Probably the most commonly-used command, `cd` allows us to change directories, that is, to jump around the file system of the operating system.

I recommend you move to your `Documents` folder. For example, on my computer, that would look like:

```

apaa2017@APAA2017-MAC21 ~ % cd Documents
apaa2017@APAA2017-MAC21 Documents % █

```

Let's say that now you want to move back to your home directory. How do you do that? You use again the `cd` command followed by two periods, which indicates one directory up. Here's how that looks on my computer:

```

apaa2017@APAA2017-MAC21 Documents % cd ..
apaa2017@APAA2017-MAC21 ~ % pwd
/Users/apaa2017
apaa2017@APAA2017-MAC21 ~ % █

```

You can experiment again with `ls` and `pwd`.

If you want to jump directly to the home directory, you can use the `~`, that is `cd ~`.

Now, type again `cd Documents` so that we are back into the `Documents` directory.

## 4.4 mkdir command

The last command we will see is `mkdir`, which creates directories. Within the `Documents` directory, we want to create a folder called `CS51`. This is where all your work for the semester should reside. Don't forget to `cd` within that newly-created folder. Next, within `cs51` create a folder called `assignments`, navigate to it using `cd` and create a new folder called `assignment1`. `cd` to `assignment1`. That's how it looks on my end:

```
apaa2017@APAA2017-MAC21 Documents % mkdir cs51
apaa2017@APAA2017-MAC21 Documents % cd cs51
apaa2017@APAA2017-MAC21 cs51 % mkdir assignments
apaa2017@APAA2017-MAC21 cs51 % cd assignments
apaa2017@APAA2017-MAC21 assignments % mkdir assignment1
apaa2017@APAA2017-MAC21 assignments % cd assignment1
apaa2017@APAA2017-MAC21 assignment1 % pwd
/Users/apaa2017/Documents/cs51/assignments/assignment1
```

And with that, you are set for the command line. Here are two more tips:

- Tab completion. Whenever you type a directory or file name, you don't need to spell out the whole thing. Just use the tab key on your keyboard. For example, typing `cd Doc` and then pressing tab should autocomplete to `cd Documents`. Great time saver!
- Up- and down-arrow. The up- and down-arrow keys on your keyboard allow you to navigate the history of the commands you have used so far. The up-arrow takes you back in time, and the down-arrow takes you forward in time.

## 5 Using the Python interpreter

If you have not seen Python before or need a refresher, you may find this brief introduction to Python syntax useful [http://cs.pomona.edu/classes/cs51/assignments/python\\_syntax\\_guide.pdf](http://cs.pomona.edu/classes/cs51/assignments/python_syntax_guide.pdf). Please note that if you are concurrently enrolled in CSCI050, you are not expected to be familiar with concepts like control flow and loops. But you should still bookmark the guide and refer to it regularly when you program. We will use it to grade the style of your code.

Ok, we are actually ready to work with Python! When working with Python, we will have two options. We can either use the Python shell, which we can think of as scratch paper that we use for quick work that we don't care about saving, or write code in files that we will save and have access to even if we close VS Code.

### 5.1 The Python shell

In your terminal, type `python3`. 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.

If all goes well, you will immediately see three right arrows appear. Go ahead and play with writing simple statements in Python. For example, I tried multiplication, exponentiation, declaring a variable `x`, adding to it, seeing its contents, using its value to declare and assign a value to a new variable `y`.

```
>>> 2*5
10
>>> 2**5
32
>>> x=2
>>> x+3
5
>>> x
2
>>> y=x-1
>>> y
1
>>> x
2
>>> █
```

Experiment on your own with the following arithmetic operators:

- Addition: `+`
- Subtraction: `-`
- Multiplication: `*`
- Division: `/`
- Integer division: `//`
- Power/exponentiation: `**`
- Mod/remainder: `%`

Remember, variables are containers we use to store values. Assignment statements link a variable name or identifier (left-hand) to a value (right-hand). It tells the interpreter to do something, but does NOT represent a value.

For example, `x = 2` declares the variable `x` and assigns to it the value 2.

If you want to work with strings, that is variables that represent text characters, you will want to enclose them in single or double quotes, e.g., `college = 'Pomona'`.

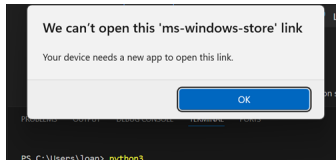
Talking about declaring variables, it's best to choose meaningful names/identifiers (so `x` is a terrible name unless you are representing an `x` coordinate). By convention, in Python, variable names should be all lowercase, and if they consist of multiple words, they should be separated by an underscore, e.g., `number_of_students`.



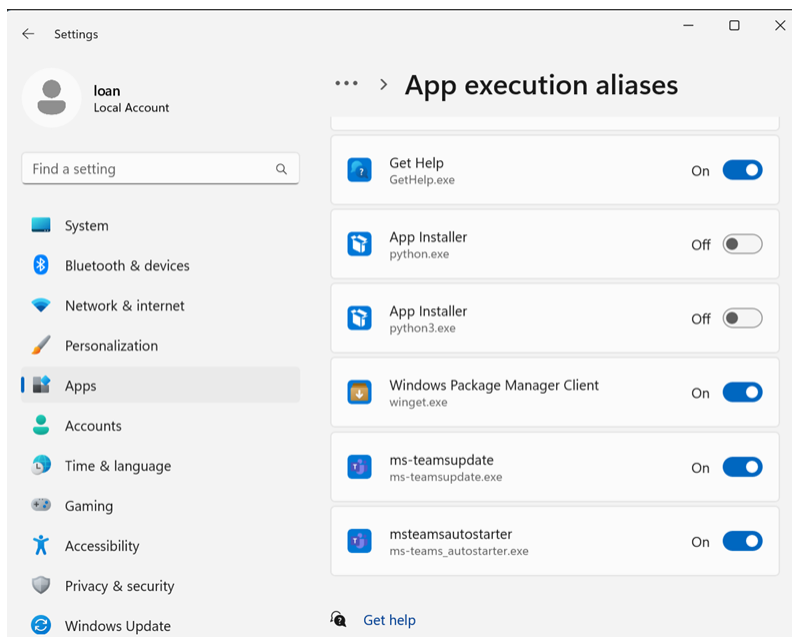
When you want to exit the Python shell, type `exit()`. If you have done something exceptionally time-consuming, e.g., you tried to calculate  $(10^{10} \cdot 1000)$  (don't!), chances are the Python shell or even your entire computer will slow down and might even crash. You can always hit **Ctrl/Command**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

## 5.2 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-windows-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\loam\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`.

```
Command Prompt
Microsoft Windows [Version 10.0.26100.1742]
(c) Microsoft Corporation. All rights reserved.

C:\Users\loan>where python
C:\Users\loan\AppData\Local\Programs\Python\Python313\python.exe

C:\Users\loan>cd C:\Users\loan\AppData\Local\Programs\Python\Python313
C:\Users\loan\AppData\Local\Programs\Python\Python313>mklink /H python3.exe python.exe
Hardlink created for 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 the course staff.

## 5.3 Creating a Python project

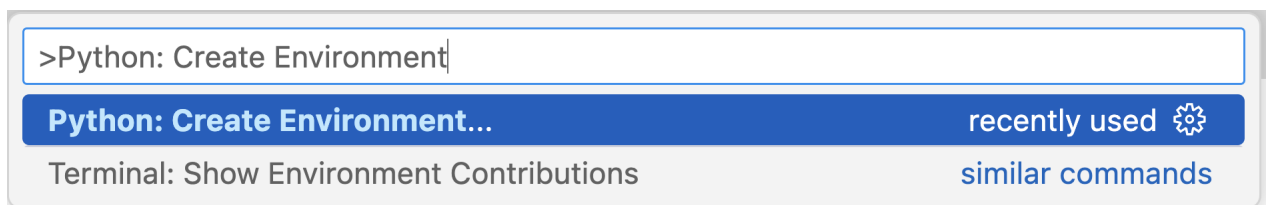
Working with the Python shell is great if we want to use Python as a calculator and for some superficial work, but our needs as programmers quickly go beyond that. We will now see how to create a Python project and write and run code in files.

### 5.3.1 Creating a virtual environment

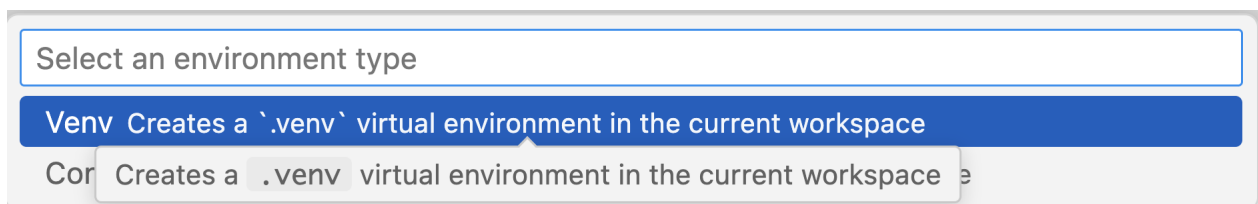
As we write more advanced Python code, we will find ourselves needing to install packages. To isolate changes to the specific projects we are working on, it is a good idea to create a *virtual environment*.

We will start by going to `File>Add Folder to Workspace` and selecting the newly created `assignment1` folder under `Documents` and `assignments`. If prompted, agree that you trust the authors in this folder.

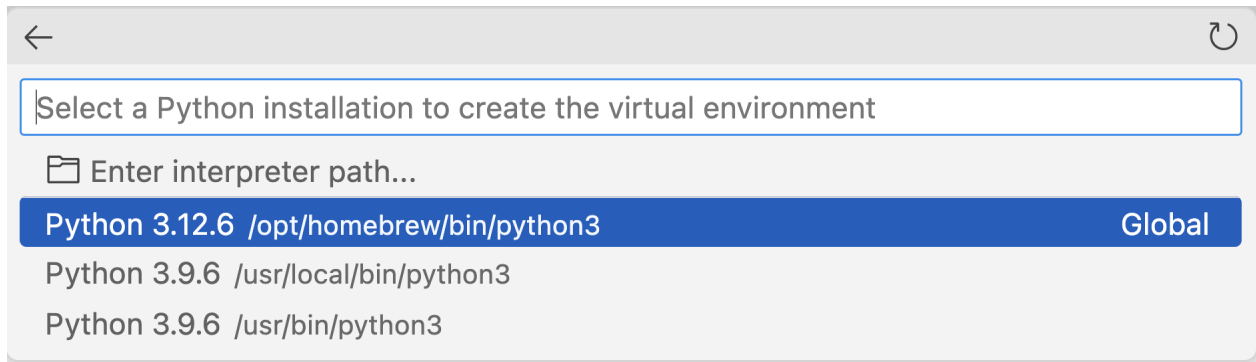
Bring up the `Command Palette` (on macOS: `Command+Shift+P`, on Windows: `Ctrl+Shift+P`) and type `Python: Create Environment`.



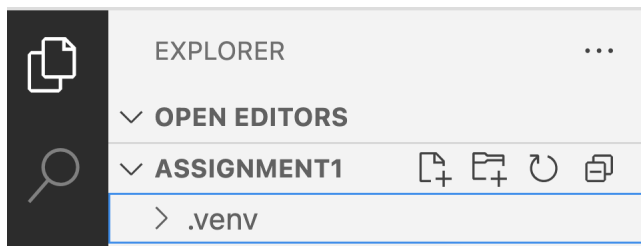
Select `venv`.



When prompted, pick the Python interpreter we installed. It should be something like `Python 3.13`.



You can confirm that the virtual environment has been selected by looking in the Explorer:

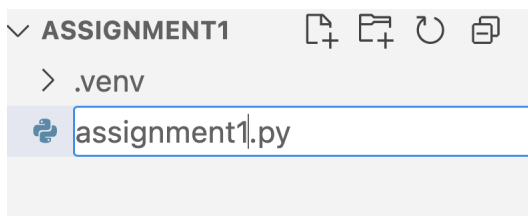


## 5.4 Creating a new Python file

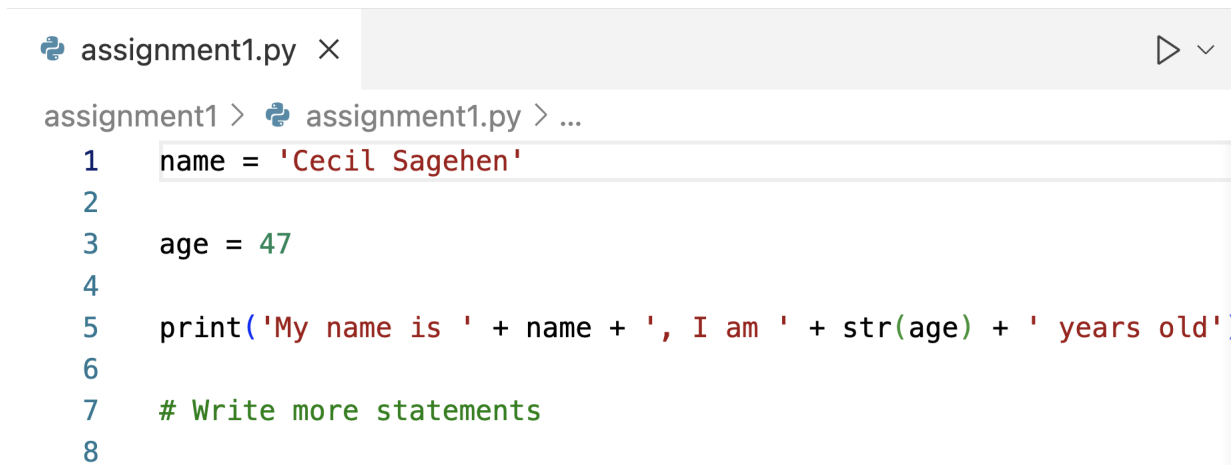
From the File Explorer toolbar, select the left-most **New File** button on the assignment1 folder (it looks like a page with a plus sign):



Name your file `assignment1.py`. 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 on the right. Edit the file to create two variables, `name` and `age`, and assign to them your name and age. Write a print statement that prints your name and age. For example, for me, that would look like:



Select Run Python File from the top right:



Looking at the terminal, I see the following printed message:

```
My name is Cecil Sagehen, I am 47 years old
```

Instead of using the VS Code Run button, you could have also typed on the terminal `python3 assignment1.py`, which is what VS Code runs behind the scenes. The results are the same!

```
(.venv) apaa2017@APAA2017-MAC21 assignment1 % python3 assignment1.py
My name is Cecil Sagehen, I am 47 years old
```

Congratulations! You have successfully written and run your first Python project.

In contrast with the Python shell mode, we cannot just type a variable and expect to see its contents. For example, we would need to type `print(age)` instead of just typing `age` when working with Python files.

**Task:** You are now asked to write at least one more statement in your `assignment1.py` file. You can use the pound sign (`#`) to write comments.

## 6 LaTeX

As computer scientists, we spend a lot of time writing code using tools like VS Code and the terminal, but we also find ourselves needing to disseminate findings in plain English or write a lot of Math. You are already familiar with document editors, such as Microsoft Word and Google Docs. Such tools are called WYSIWYG (what you see is what you get) and allow you to typeset your text by creating sections, bolding text, etc., using a graphical interface.

Rather than using such tools, computer scientists love LaTeX (pronounced “lah-tech”, with a soft ch). LaTeX files (their extension is `.tex`) contain plain text and special commands that signify

the typesetting. This very document has been written using LaTeX, and the next step of this assignment is to introduce LaTeX and get you to write your first document in it.

For this, we will use a website called Overleaf. You should go ahead and sign up (feel free to use either your Pomona or personal email account, just make sure you save your password). This will allow you to create LaTeX projects for free. You can also add up to one collaborator, which will be sufficient for this class when we have pair assignments.

Once you have an account, please go ahead and complete Overleaf's LaTeX 30-minute tutorial.

## 6.1 Writing your own LaTeX file

Now that you have learned about LaTeX, we ask you to go to Overleaf and create a new project called `CS51-assignment1`. Create a new file called `assignment1.tex`.

Your document should answer the following questions:

- What name would you like to be called? Do you want to tell us your personal pronoun? Is there anything else I should know?
- Where are you from? Where did you go to high school?
- Do you have any prior programming experience?
- Are you concurrently enrolled in CSCI050?
- What are your academic interests and/or planned major? Do you have a long-term goal or dream job in mind for the future?
- What do you enjoy doing in your free time?
- What do you value the most?
- Which aspects of this course are you most looking forward to?
- Which aspects of this course are you most nervous about?
- What stood out to you about what we covered this week, either on the history or ethics of computer science?
- Is there anything else you'd like us to know?

We ask you to answer these questions, but you can improvise in how you want your LaTeX file to look. Your document must at a minimum include the following:

- A title section containing a `\title`, `\author`, and `\date`. The title should be *Assignment 1*, the author should include your full name, and the date should automatically generate today's date.

- At least one `\section`
- A figure (maybe a picture of yours or something you find relevant to history or ethics of computer science)
- A table
- A list
- Text that references (`\ref`) each of your figures and tables.

Once you are done, download the source, which will include the `assignment1.tex` file and the `assignment1.pdf`.

## 7 Feedback

Using VS Code, create a file named `feedback1.txt` (in the `assignment1` folder) that answers the questions:

- How long did you spend on this assignment?
- Any comments or feedback? What things did you find interesting, challenging, or boring?

## 8 When you're done

Submit your `assignment1.py`, `assignment1.tex` and resulting `assignment1.pdf`, and `feedback1.txt` files online using Gradescope; be sure you're uploading them to the right assignment. Note that you can resubmit the same assignment as many times as you would like up until the deadline.

## 9 Grading

	points
Successfully logged in a lab machine	1
Submitted the four required files on Gradescope	1
Wrote at least one additional Python statement in <code>assignment1.py</code>	1
Reflected on all questions for <code>feedback1.txt</code>	1
<code>assignment1.tex</code> contains a title,	1
at least one section,	1
a figure,	1
a table,	1
a list,	1
and text that references each of the figures and tables	1
Total	10