



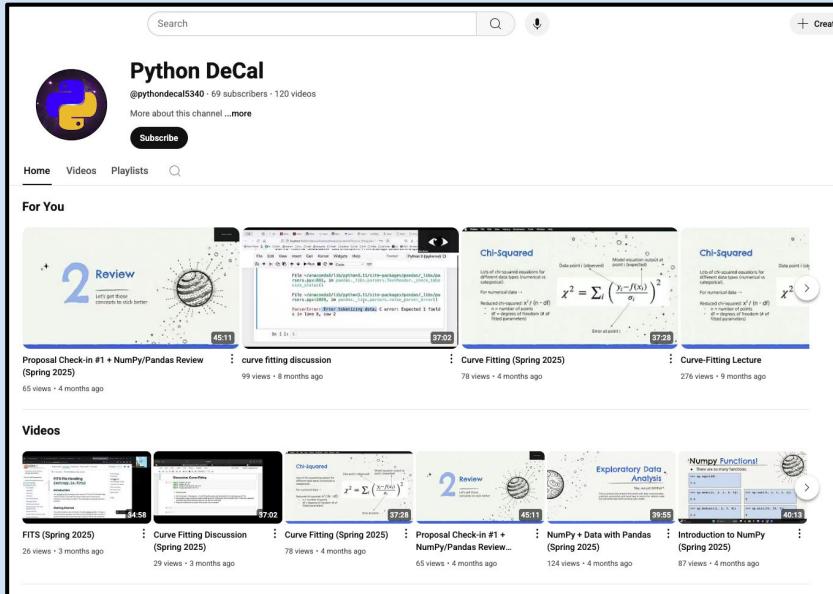
wELCOME TO THE PYTHON DECAL!

VS Code + Variables + Data Types + Command Line/Terminal
[Class 2] September 3rd, 2025



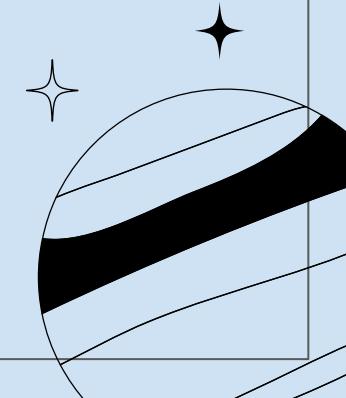


LECTURES WILL BE RECORDED



This class will be
recorded and posted to
our YouTube Channel!

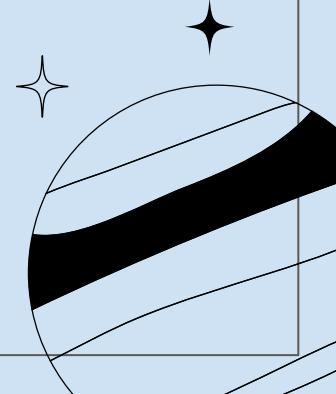
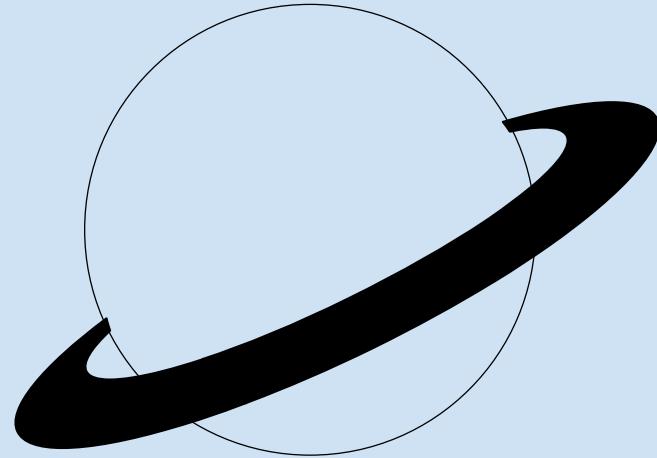
Subscribe :)



C

OVERVIEW FOR TODAY!

1. Announcements
2. Super Quick Warm-Up
3. Demo
 - a. VS Code (+ Comments)
 - b. Variables + Data Types
 - c. Practice Question?
 - d. Command Line/Terminal
4. Lecture Engagement
5. Bonus Content (for homework help)





* 01 *

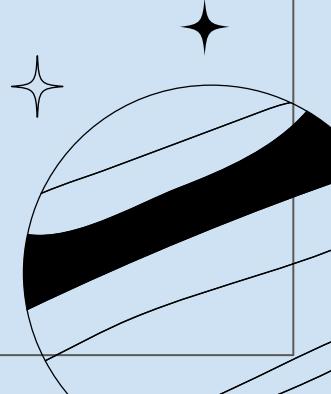
Announcements

:) :) :) :) :) :) :) :) :)):



Announcements

- HW #1 was released on Monday, Jan. 26th
- HW #1 is due next Wednesday, Feb. 4th at 11:59 PM
- HW #1 late due date is Feb. 18th at 11:59 PM
 - Next, next, next Wednesday
- Office hours w2m has been posted on bCourses + Ed
 - Please fill this out by Friday EOD!
- Please finish up the installation guide if you haven't yet
 - Installation office hours throughout this week





* 02 *

Warm-Up

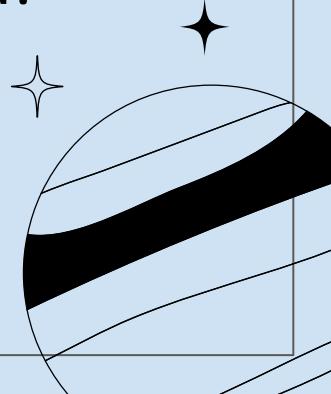
Let's do a little review



SUPER QUICK WARM-UP

Answer the following questions with a partner:

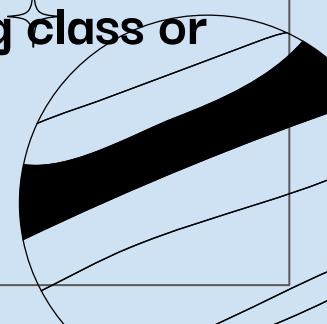
- 1) What days are homeworks released? When are they due?
- 2) Where are homeworks posted? Where do you turn them in?
- 3) When is the late due dates for homework?
- 4) What are lecture engagements? Where/when are they due?
- 5) Where are slides posted? Where are recordings posted?
- 6) What do you need to pass this class?
- 7) When are Final Project Presentations?
- 8) What are some ways to get help?



C

warm-UP Answers

- 1) Released on Mondays. Due the following Wednesday
- 2) Posted on bCourses. Turn in on Gradescope.
- 3) Two Wednesdays past the due date.
- 4) Participation for class. Due on bCourses before the next class.
- 5) Slides posted on bCourses. Recordings posted on YouTube.
- 6) Overall grade of 70% and an attempt on all Final Project assignments.
- 7) Last week of instruction! (the week before dead week)
- 8) Office hours! Ed discussion, email, asking questions during class or after class, Google

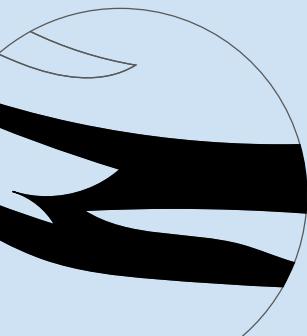




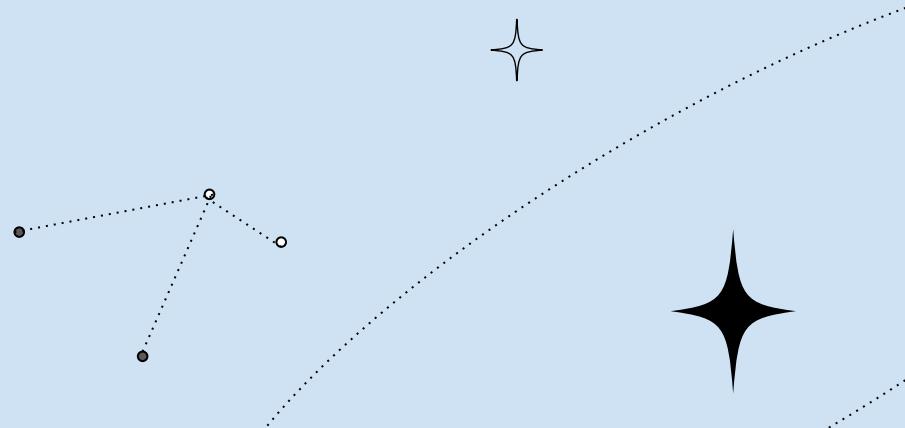
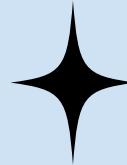
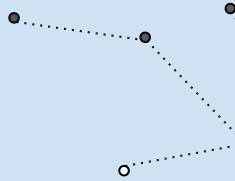
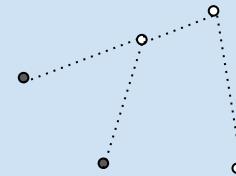
* 03 *

Demo

Starting with VS Code



VS CODE



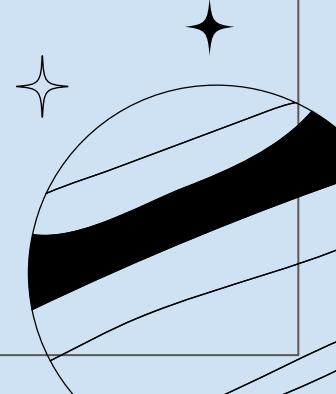


WHAT IS VS CODE?

VS Code is one of the two main applications we will use to write and edit code this semester.

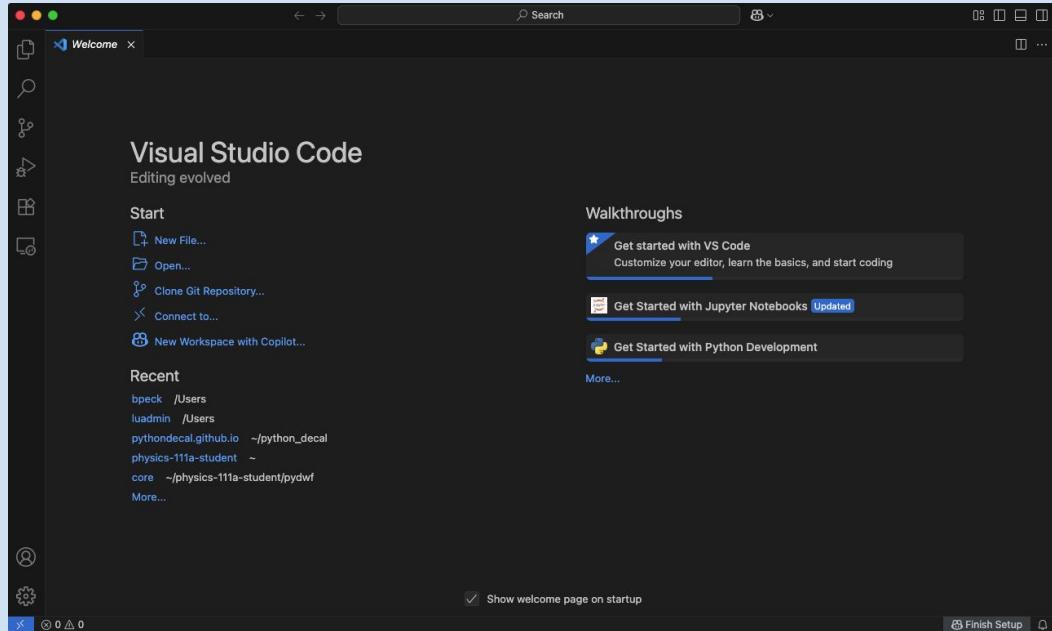
Why do we use it?

- It's open source (free)
- Lots of features and very customizable
- Works well with the terminal





OPEN VS CODE



Search for VS Code under Applications or Downloads

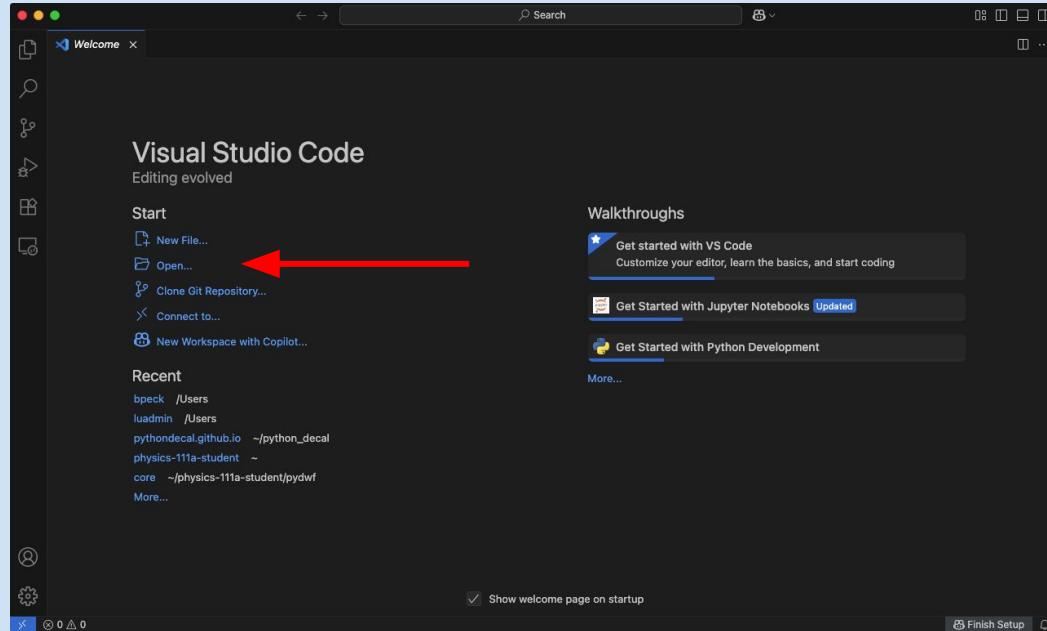
Open VS Code

- This is what your screen should look like when its open





OPEN A NEW FOLDER IN VS CODE

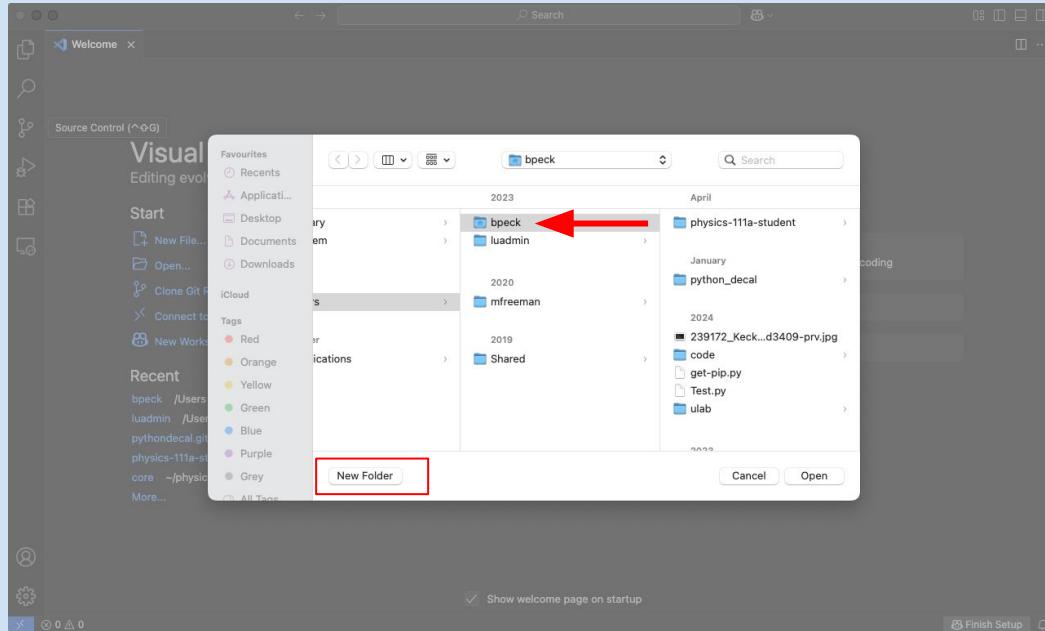


Click on Open...

We will be creating a new folder for this class

You will work primarily in this folder for the whole semester

OPEN A NEW FOLDER (CONT.)

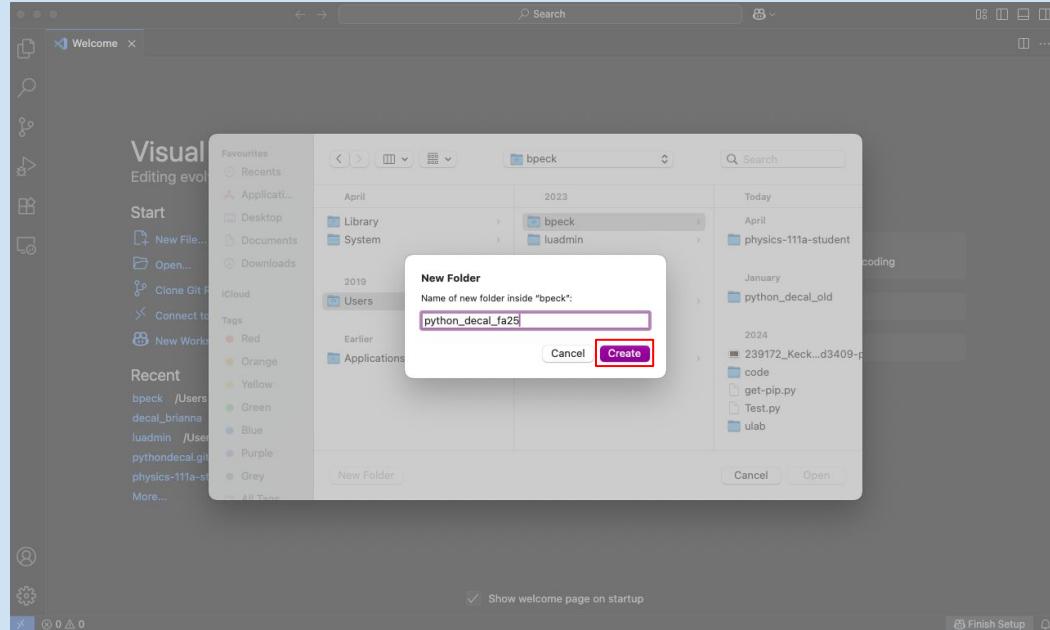


Open a folder in the default location under your user

So I am opening a new folder under bpeck because that is my user



OPEN A NEW FOLDER (CONT.)



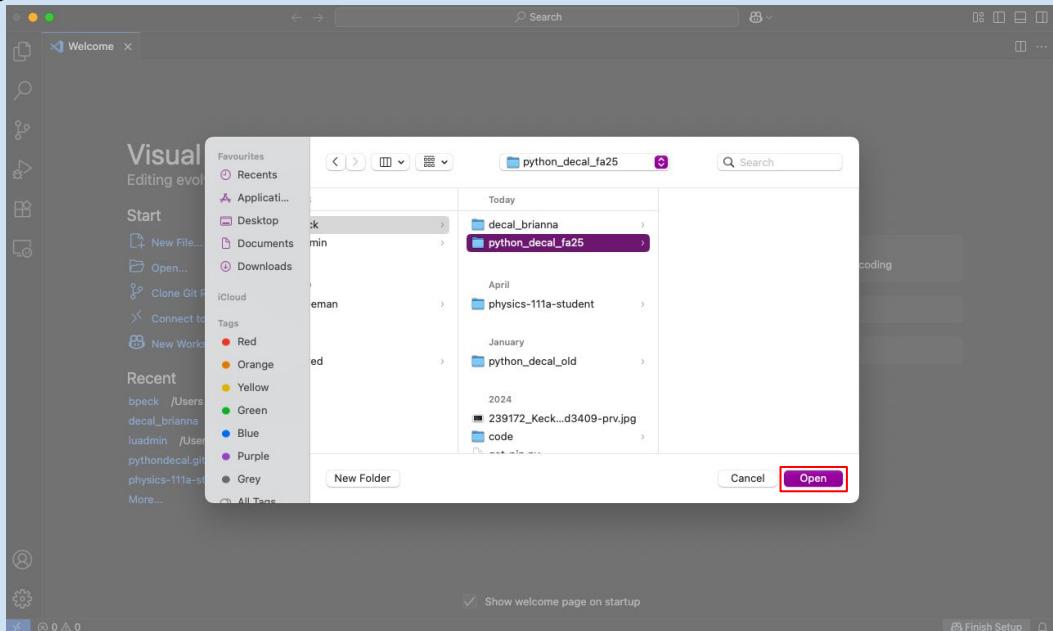
Call your new folder:
python_decal_sp26

In this class, we will use
folder and directory
interchangeably

They mean the exact
same thing



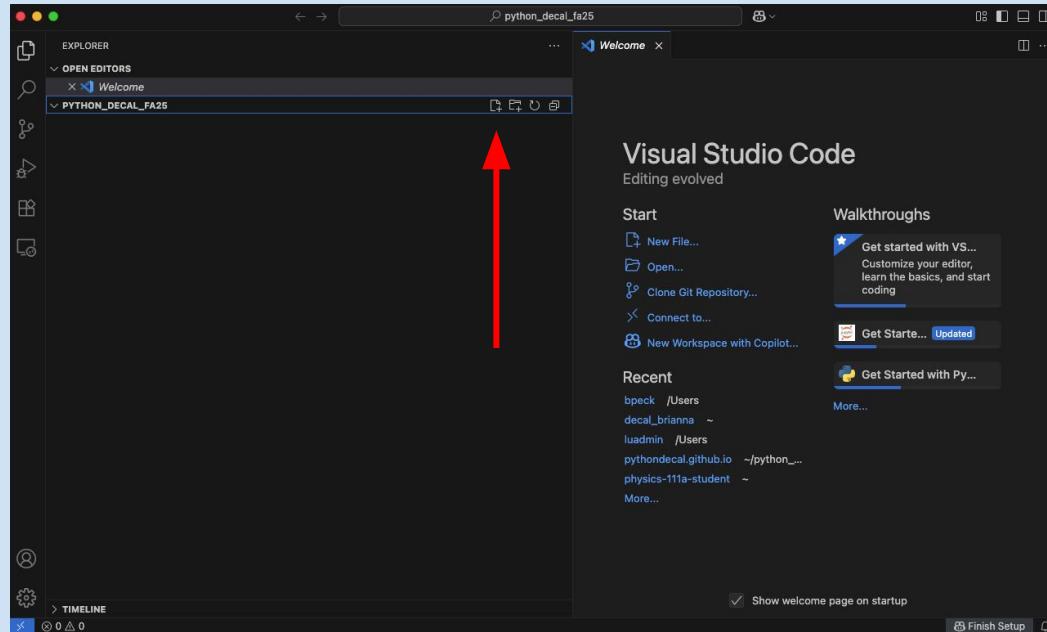
OPEN A NEW FOLDER (CONT.)



Open up your new
python_decal_sp26
directory

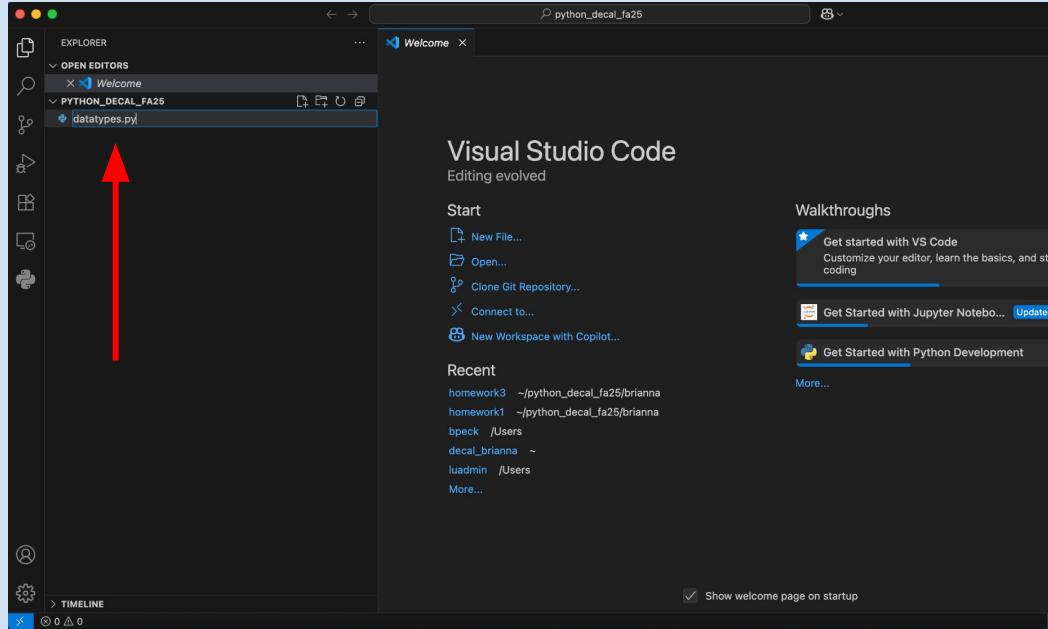
The last column should be
completely empty since
there is nothing in this
folder yet.

CREATE A NEW PYTHON FILE



To create a new file,
click on the icon the red
arrow is pointing to

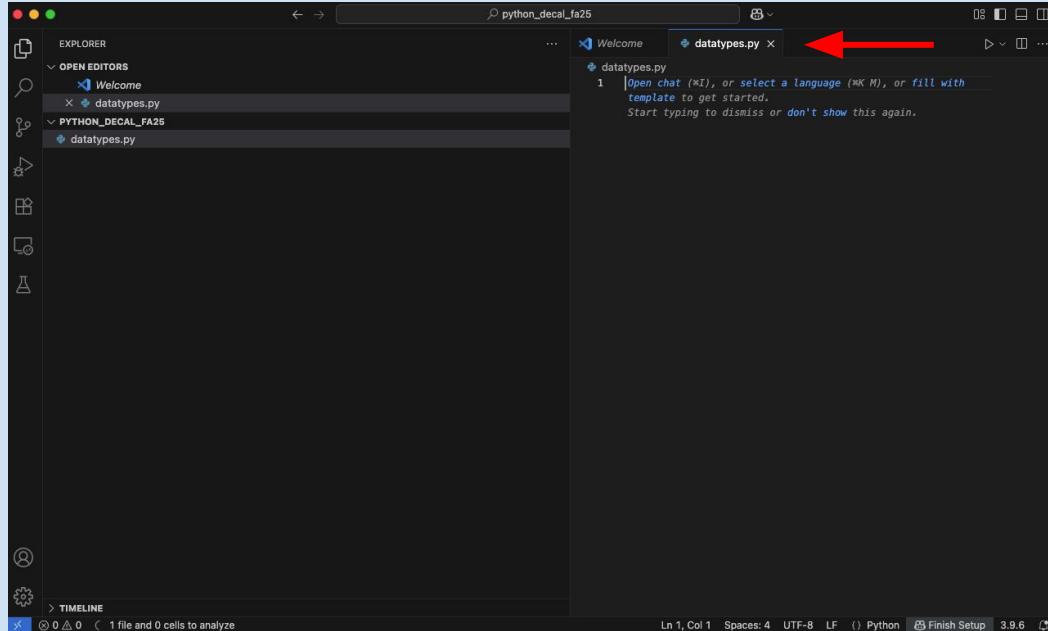
create a new FILE (cont.)



Under the new folder,
name your new file:
datatype.py

The first topic we will
cover today (besides
VS Code) are data
types

create a new FILE (cont.)

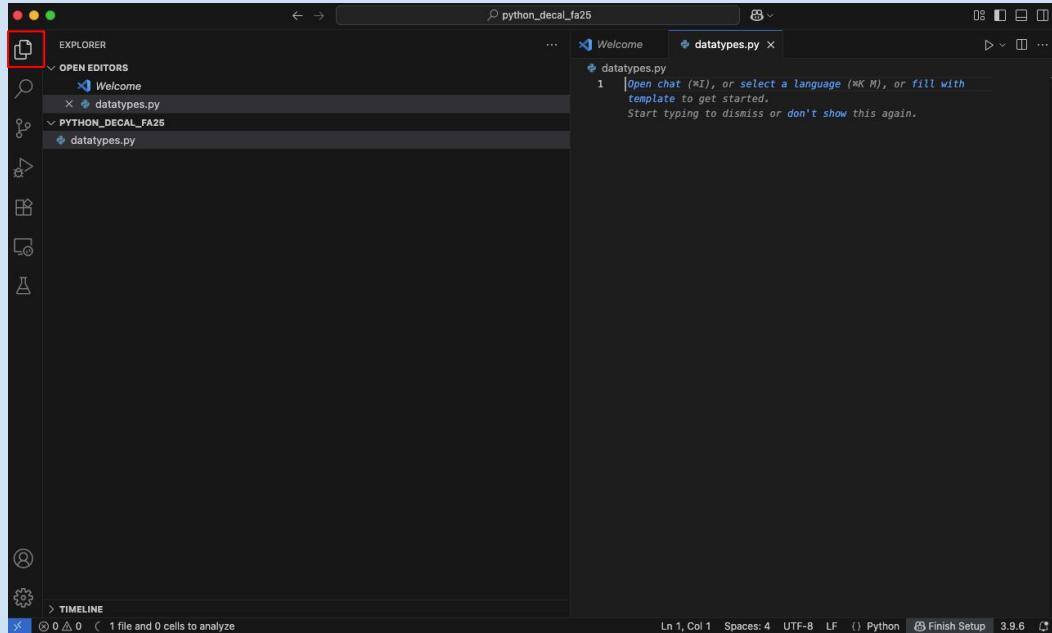


On the right hand side of your window, a new tab will appear

You should see the name of your file at the top of the window



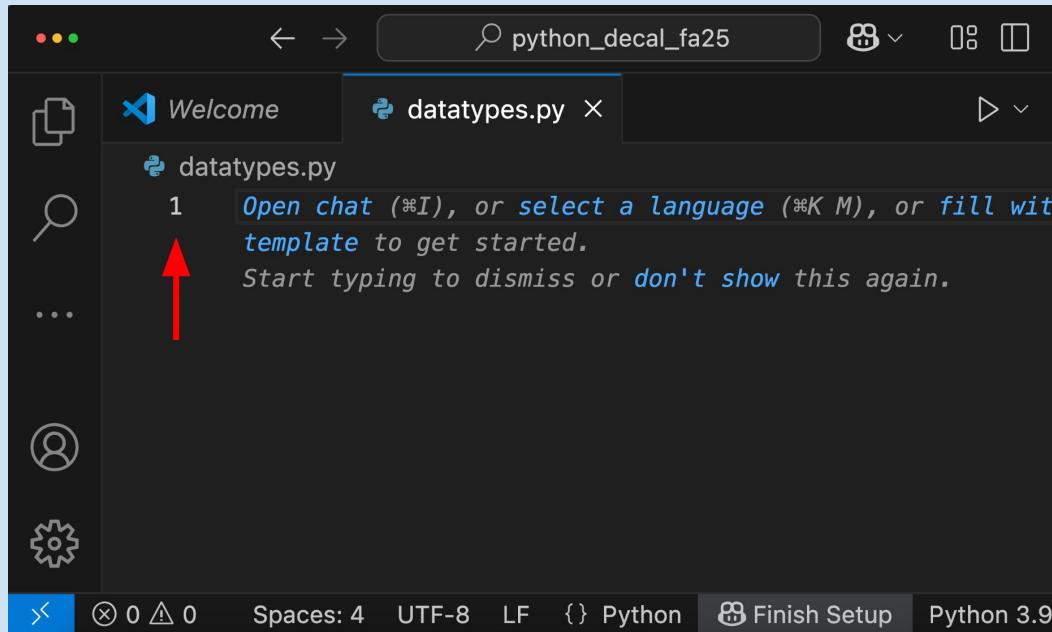
create a new FILE (cont.)



To close the window on the left (Explorer), click the icon in the red box



write a PYTHOn SCRIPT



Script = a file containing python code that can perform tasks

Time to write our first python script!

Click on the line next to the number 1



write a PYTHON script (cont.)

A screenshot of a dark-themed code editor window. The title bar says "python_decal_fa25". The left sidebar has icons for file, search, and settings. The main area shows a file named "datatypes.py" with one line of code: "print("Hello, World!")". The status bar at the bottom shows "Spaces: 4" and "Python 3.9".

```
python_decal_fa25
Welcome datatypes.py
datatypes.py
1 print("Hello, World!")
```

On line 1, type:
print("Hello, World!")

This is instructing Python
to “print” the words:
Hello, World!

The words need quotation
marks on **both sides** of the
phrase



write a PYTHON SCRIPT (CONT.)

A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows the search term "python_decal_fa25". The left sidebar has icons for file explorer, search, and other settings. The main area shows a Python file named "datatypes.py" with the following code:

```
1 print("Hello, World!")
```

The status bar at the bottom indicates "Spaces: 4" and "Python 3.9". A red arrow points to the status bar, highlighting the small white dot next to the file name "datatypes.py", which serves as an indicator for unsaved changes.

A white dot will now appear next to the top of your file

This means you have unsaved changes

Call: Ctrl + S or Cmd + S to save your changes



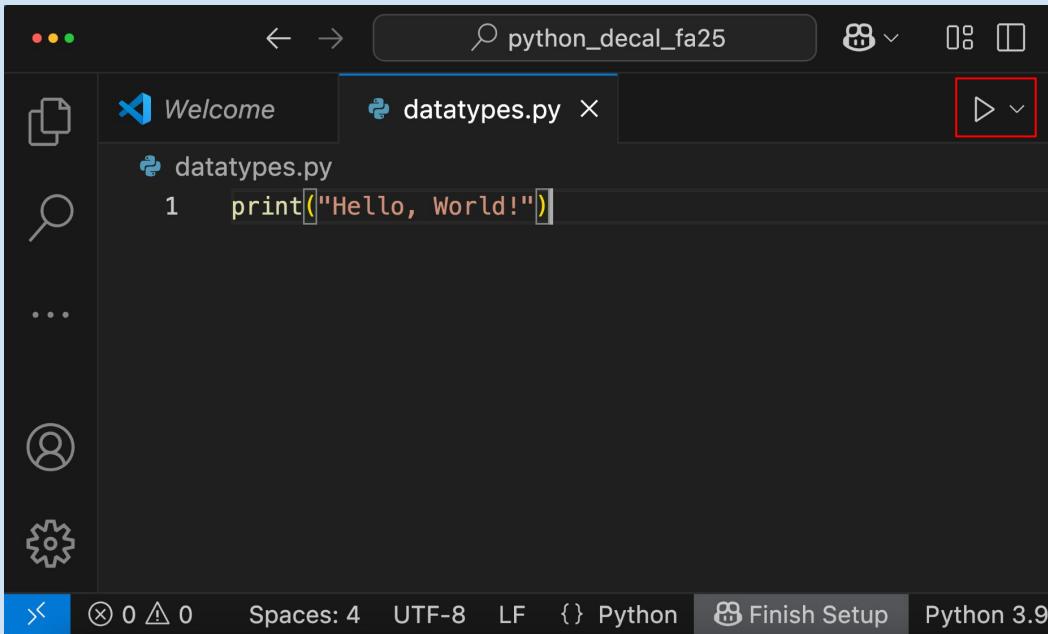
write a PYTHON SCRIPT (CONT.)

A screenshot of a dark-themed code editor window. The title bar shows the search term "python_decal_fa25". The left sidebar has icons for file, search, and settings. The main area shows a file named "datatypes.py" with the code "print('Hello, World!')". A red arrow points to the file tab. The bottom status bar shows "Spaces: 4" and "Python 3.9".

This is what the symbol
will look like when you
have saved your changes

It's always good to save
your work after you make
changes

RUN A PYTHON SCRIPT



A screenshot of a Python code editor interface. The title bar shows the search term "python_decal_fa25". The left sidebar has icons for file, search, and settings. The main area shows a file named "datatypes.py" with the following content:

```
1  print("Hello, World!")
```

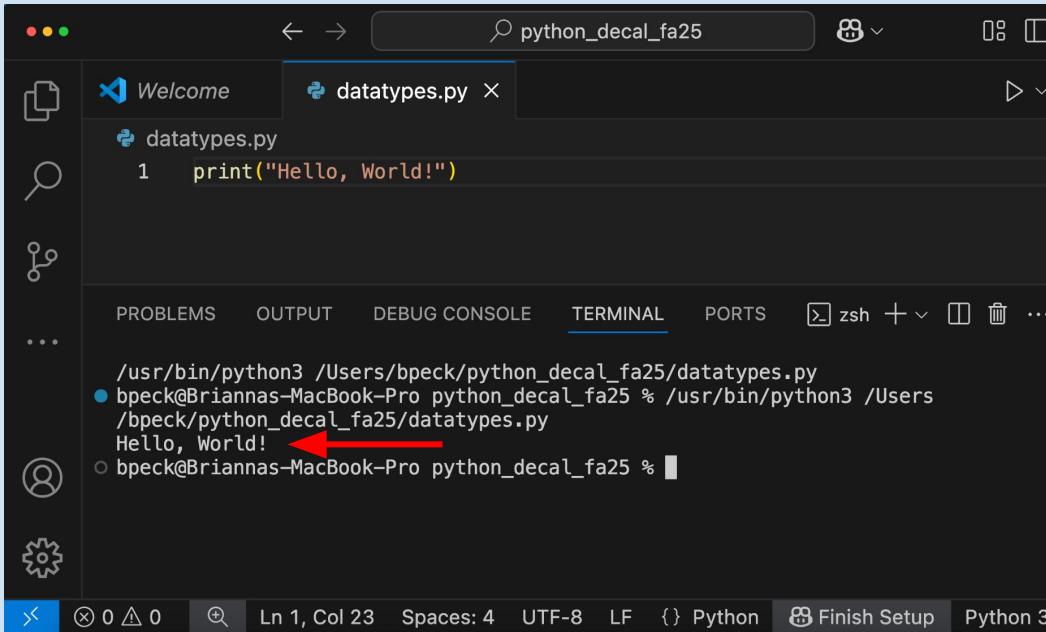
The status bar at the bottom shows "Spaces: 4" and "Python 3.9".

To run your python script,
click on the arrow in the
red box

Run a script = telling your
computer to execute the
program you created

Scripts go line-by-line

RUN A PYTHON SCRIPT



A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows the search term "python_decal_fa25". The left sidebar has icons for file, search, and user. The main area shows a file named "datatypes.py" with the code:

```
1  print("Hello, World!")
```

The bottom right corner of the code editor has a red arrow pointing to the terminal tab. The terminal tab is active and displays the command-line output:

```
/usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py
● bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users
/bpeck/python_decal_fa25/datatypes.py
Hello, World!
```

The status bar at the bottom shows: Ln 1, Col 23, Spaces: 4, UTF-8, LF, Python, Finish Setup, Python 3.

A new window will appear on the bottom of your screen

You should see the words:
Hello, World!

This window is called the terminal, it shows the output of your code



RUN A PYTHON SCRIPT



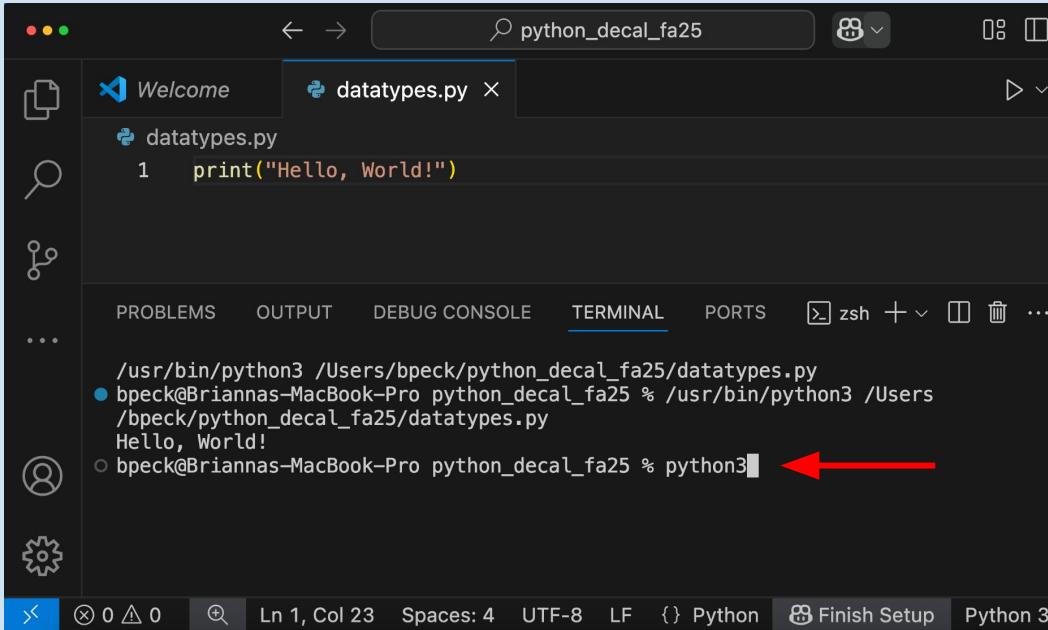
A screenshot of the Visual Studio Code interface. The title bar shows the file name "python_decal_fa25". The left sidebar has icons for File, Find, and Settings. The main area shows a Python file "datatypes.py" with the single line of code: "print('Hello, World!')". Below the editor is a terminal window titled "zsh" showing the command and its output: "/usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py" followed by "Hello, World!". A red arrow points to the terminal prompt "%". The bottom status bar shows "Ln 1, Col 23" and "Python 3".

VS Code allows you to access your files and the terminal at the same time

This percentage % symbol, means that your terminal is awaiting your next command.



PYTHON DIRECTLY IN TERMINAL



```
python_decal_fa25
```

```
Welcome datatypes.py
```

```
datatypes.py
1 print("Hello, World!")
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL zsh + ...
```

```
/usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py
Hello, World!
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3
```

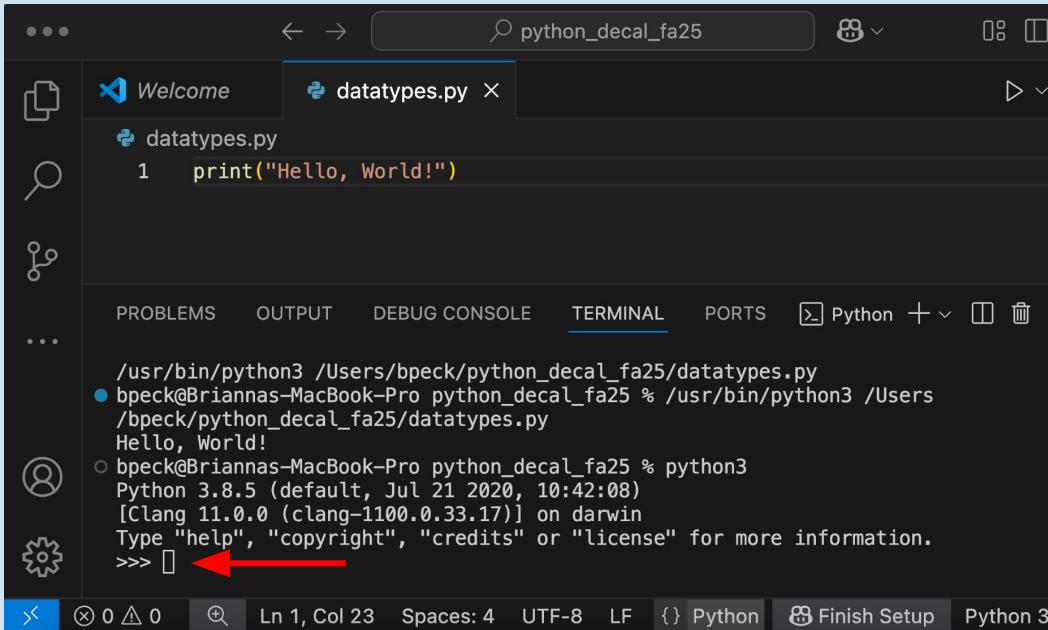
The screenshot shows a dark-themed interface of VS Code. In the center, there's a code editor with a file named 'datatypes.py' containing a single line of code: 'print("Hello, World!")'. Below the editor is a terminal window. The terminal has tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL', with 'TERMINAL' being the active tab. It displays the command '/usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py' followed by the output 'Hello, World!'. At the bottom of the terminal, the command 'python3' is partially typed, with a red arrow pointing to the cursor position.

In addition to running scripts, you can type Python code directly on your terminal

Open Python by typing:
python3 into your
command line

Then press Return/Enter

PYTHON DIRECTLY IN TERMINAL

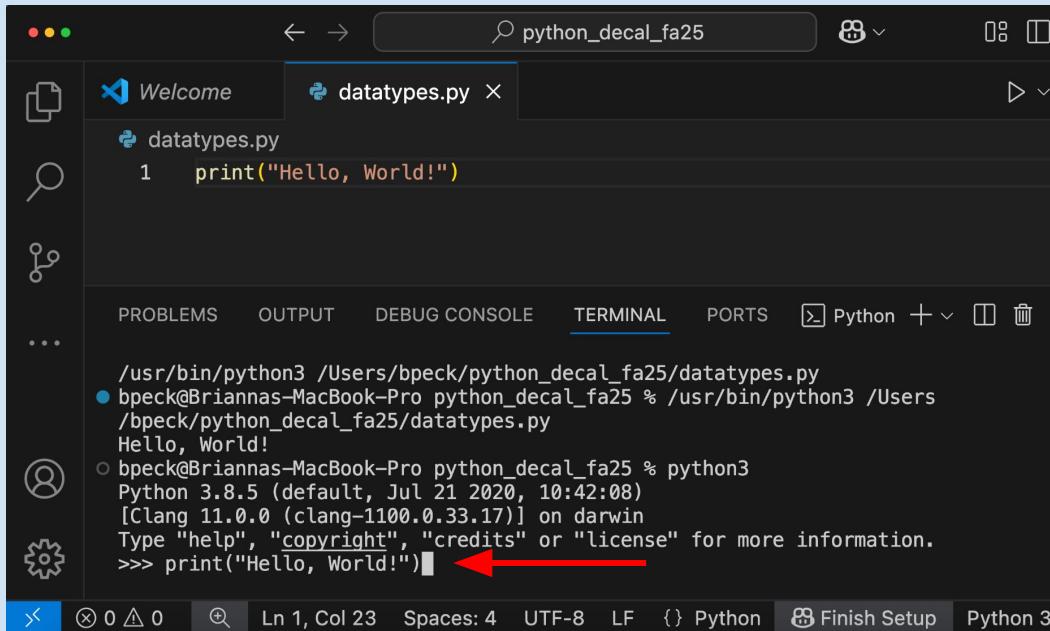


A screenshot of the Visual Studio Code interface. The left sidebar shows icons for file, search, and settings. The main area has tabs for 'Welcome' and 'datatypes.py'. The code editor shows a single line of Python code: 'print("Hello, World!")'. Below the editor is a tab bar with 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', 'PORTS', and a Python icon. The 'TERMINAL' tab is selected. The terminal window displays the output of running the script: '/usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py' followed by 'Hello, World!'. A red arrow points to the 'Python' icon in the tab bar.

Now your terminal now
“speaks” in the Python
language instead of
zsh/bash!

These three arrows >>>,
mean the terminal is
speaking in Python

RUN PYTHON IN THE TERMINAL



A screenshot of a terminal window titled "python_decal_fa25". The window shows a file named "datatype.py" with the contents:

```
1 print("Hello, World!")
```

The terminal tab shows the output of running the script:

```
/usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py
● bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users
/bpeck/python_decal_fa25/datatypes.py
Hello, World!
```

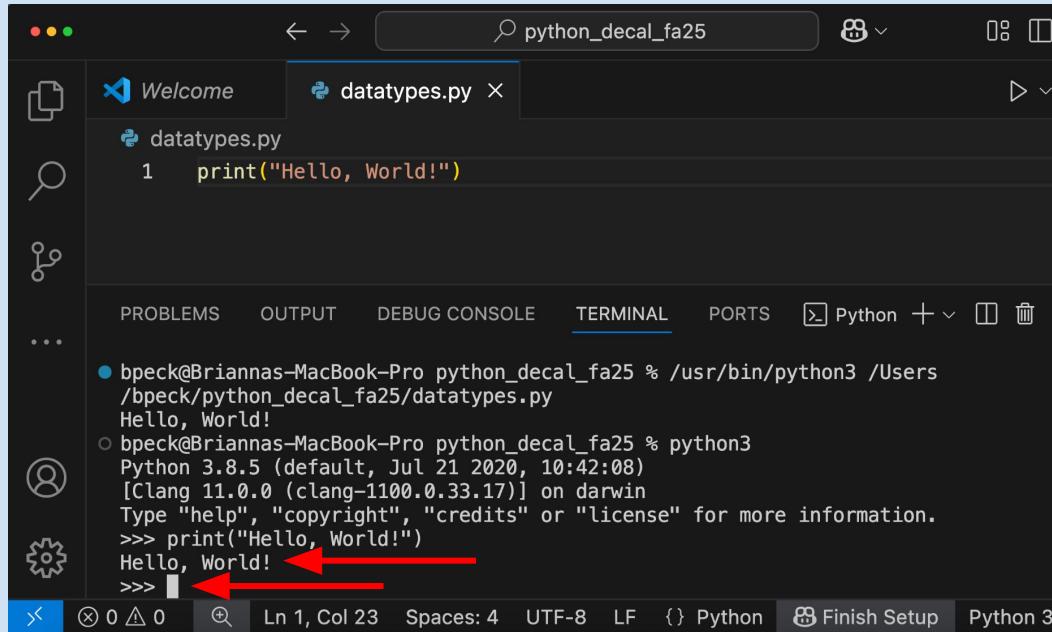
Below this, another session starts with:

```
○ bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3
Python 3.8.5 (default, Jul 21 2020, 10:42:08)
[Clang 11.0.0 (clang-1100.0.33.17)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
```

The command `>>> print("Hello, World!")` is typed into the terminal, indicated by a red arrow pointing to the cursor position.

Next to those three little arrows, type the same statement as above:
print("Hello, World!")

RUN PYTHON IN THE TERMINAL



```
python_decal_fa25
```

```
datatypes.py
```

```
1 print("Hello, World!")
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python
```

```
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py
Hello, World!
```

```
o bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3
Python 3.8.5 (default, Jul 21 2020, 10:42:08)
[Clang 11.0.0 (clang-1100.0.33.17)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
```

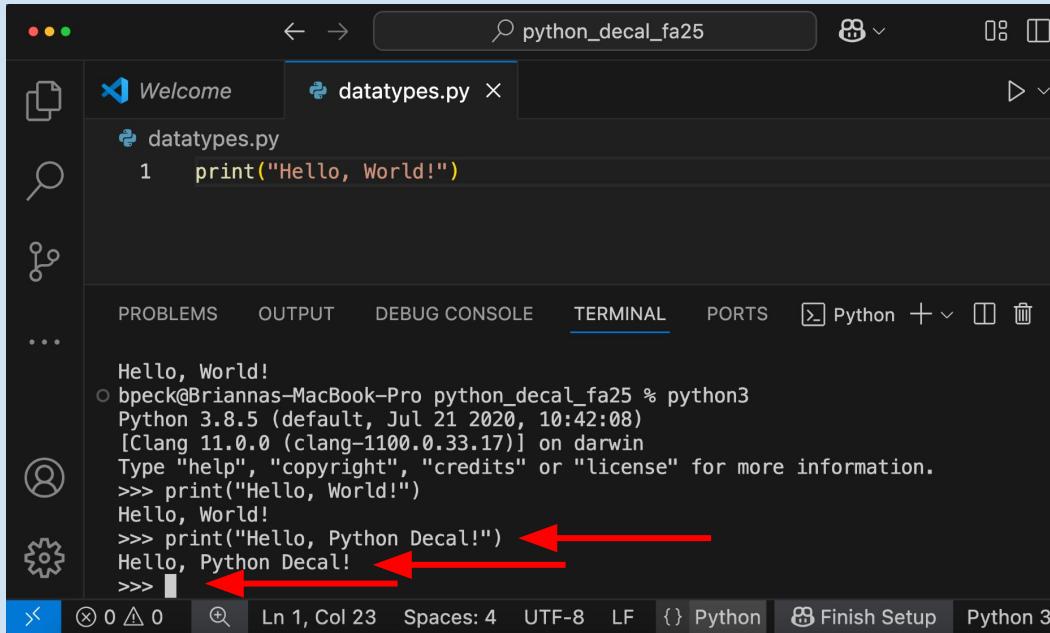
```
>>> 
```

Ln 1, Col 23 Spaces: 4 UTF-8 LF {} Python Finish Setup Python 3.

Once you hit
Return/Enter, you should
see another line of:
Hello, World!

Then three more arrows
will appear, awaiting your
next command

RUN PYTHON IN THE TERMINAL



```
python_decal_fa25
```

```
Welcome datatypes.py
```

```
datatypes.py
```

```
1 print("Hello, World!")
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python +
```

```
Hello, World!
```

```
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3
```

```
Python 3.8.5 (default, Jul 21 2020, 10:42:08)
```

```
[Clang 11.0.0 (clang-1100.0.33.17)] on darwin
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> print("Hello, World!")
```

```
Hello, World!
```

```
>>> print("Hello, Python Decal!")
```

```
Hello, Python Decal!
```

```
>>> █
```

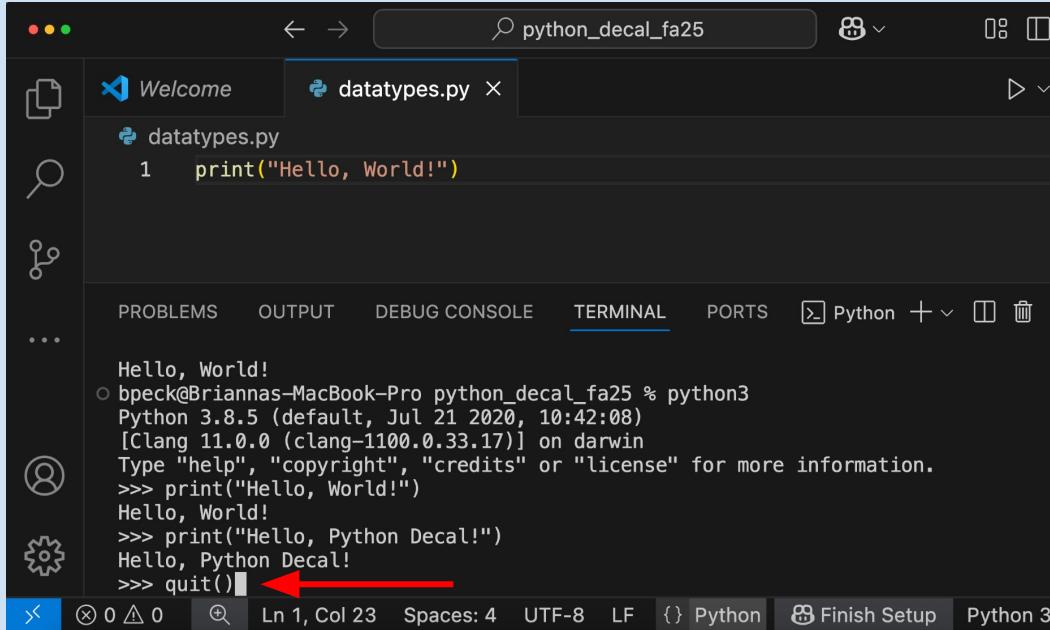
```
Ln 1, Col 23 Spaces: 4 UTF-8 LF {} Python Finish Setup Python 3
```

Let's practice one more time, by typing:
print("Hello, Python Decal!")

Then run the code!

Once your code is done, the >>> will appear again

QUIT PYTHON IN THE TERMINAL



A screenshot of the Visual Studio Code interface. The left sidebar shows icons for file operations like Open, Save, Find, and Settings. The main area has tabs for 'Welcome' and 'datatypes.py'. The code editor shows a single line of Python code: 'print("Hello, World!")'. Below the editor is a terminal window with the following output:

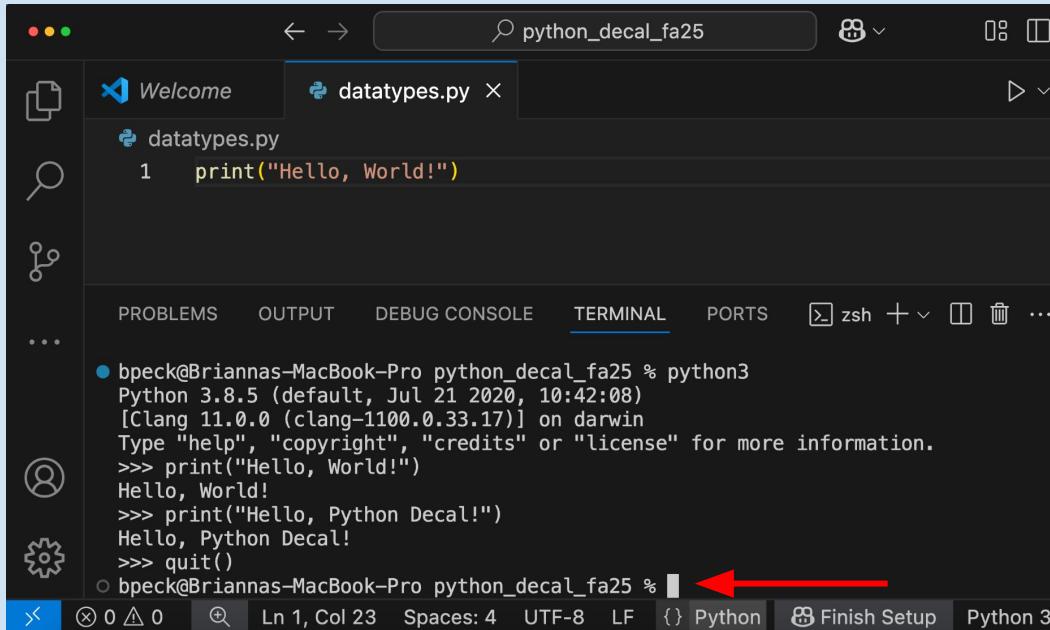
```
Hello, World!
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3
Python 3.8.5 (default, Jul 21 2020, 10:42:08)
[Clang 11.0.0 (clang-1100.0.33.17)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
>>> print("Hello, Python Decal!")
Hello, Python Decal!
>>> quit()
```

A red arrow points to the 'quit()' command at the bottom of the terminal window.

To go back to your
“zsh/bash” speaking
terminal, type:
quit()

This will close your
in-terminal Python session

BACK TO NORMAL TERMINAL



A screenshot of the Visual Studio Code interface. The title bar shows the workspace name "python_decal_fa25". The left sidebar has icons for file, search, and other tools. The main area shows a Python file named "datatype.py" with the code "print("Hello, World!")". Below the editor is a tab bar with "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", "PORTS", and "zsh". The "TERMINAL" tab is selected, displaying a terminal session:

```
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3
Python 3.8.5 (default, Jul 21 2020, 10:42:08)
[Clang 11.0.0 (clang-1100.0.33.17)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
>>> print("Hello, Python Decal!")
Hello, Python Decal!
>>> quit()
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

A red arrow points to the percentage symbol (%) at the end of the command line in the terminal.

You know you are back to the regular terminal when you see the percentage % symbol again



TERMINAL NO SPEAK PYTHON



```
python 3.8.5 (default, Jul 21 2020, 10:42:08)
[Clang 11.0.0 (clang-1100.0.33.17)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
>>> print("Hello, Python Decal!")
Hello, Python Decal!
>>> quit()
bpeck@Briannas-MacBook-Pro python_decal_fa25 % print("Hello!")
```

Now, if you try to type in Python, the terminal won't understand

For example:
print("Hello!")

Now you get a weird dquote>, because the terminal is confused



TERMINAL NO SPEAK PYTHON



A screenshot of the Visual Studio Code interface. The title bar shows the workspace name: "python_decal_fa25". The left sidebar has icons for Welcome, datatypes.py (the active file), and other tools. The main area shows a Python terminal session:

```
[Clang 11.0.0 (clang-1100.0.33.17)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
>>> print("Hello, Python Decal!")
Hello, Python Decal!
>>> quit()
bpeck@Briannas-MacBook-Pro python_decal_fa25 % print("Hello!")
```

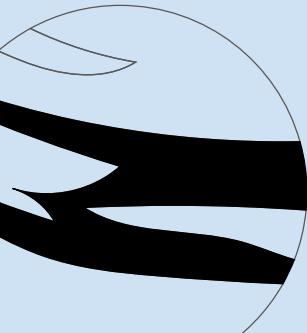
The status bar at the bottom indicates the terminal is using Python 3.

To get back to the normal terminal again, call:
Ctrl + C

This will pop back up the percentage % symbol, which awaits another command



comments





COMMENTING OUT CODE



```
python_decal_fa25
```

```
1 # print("Hello, World!")
```

```
[Clang 11.0.0 (clang-1100.0.33.17)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
>>> print("Hello, Python Decal!")
Hello, Python Decal!
>>> quit()
bpeck@Briannas-MacBook-Pro python_decal_fa25 % print("Hello!")
dquote>
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh + ▾

Ln 1, Col 25 Spaces: 4 UTF-8 LF {} Python Finish Setup Python 3.

If you want Python to ignore a line of your code, you use comments

To make a comment, add a pound # symbol at the front of the code you want commented out

The text turns green





COMMENTING OUT CODE



```
python_decal_fa25
```

datatype.py

```
1 # print("Hello, World!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh + ...

```
>>> print("Hello, World!")
Hello, World!
>>> print("Hello, Python Decal!")
Hello, Python Decal!
>>> quit()
○ bpeck@Briannas-MacBook-Pro python_decal_fa25 % print("Hello!")
dquote>
● bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py
○ bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

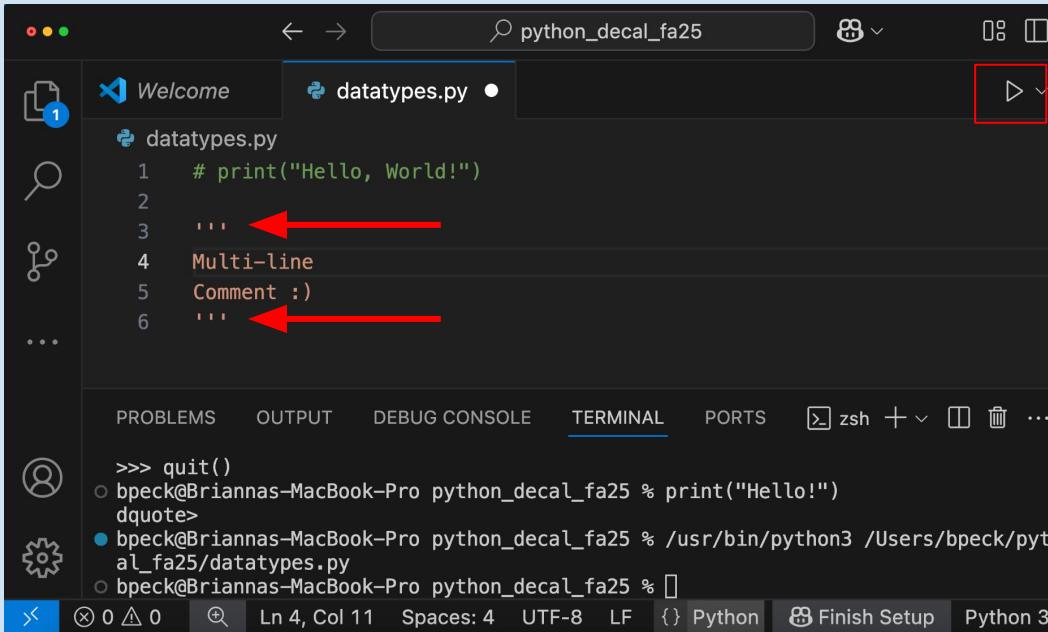
Ln 1, Col 25 Spaces: 4 UTF-8 LF {} Python Finish Setup Python 3.

If we press the arrow at the top of the window, now the Terminal will not return anything

Comments are for human eyes only, they are extremely useful! More on that later



MULTI-LINE comments



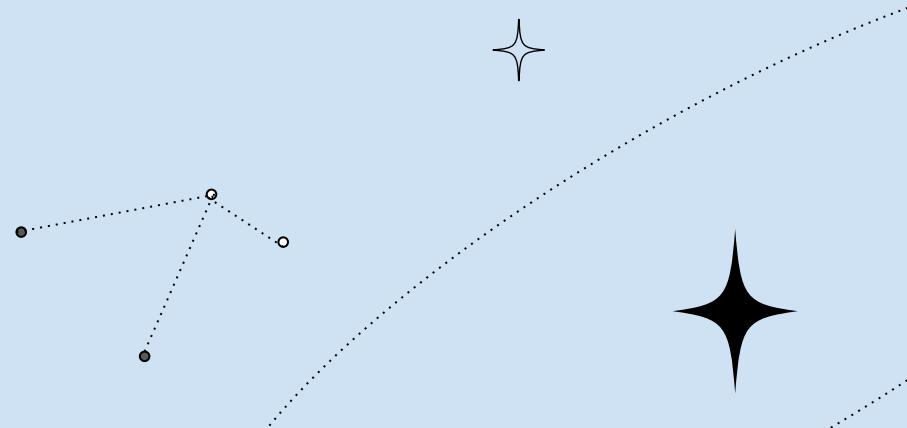
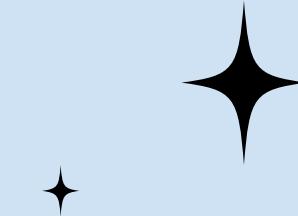
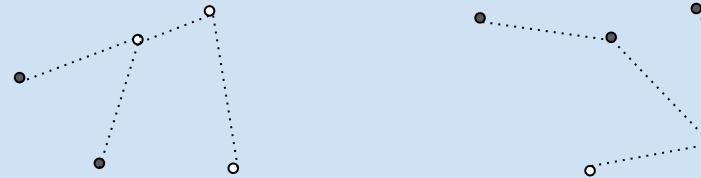
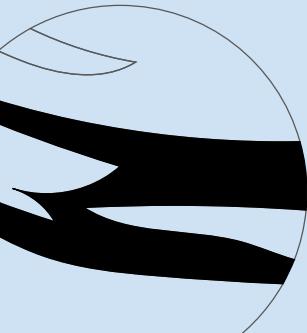
```
# print("Hello, World!")  
'''  
Multi-line  
Comment :)  
'''
```

The screenshot shows a dark-themed code editor window for VS Code. The title bar says "python_decal_fa25". The left sidebar has icons for file, search, and settings. The main area shows a Python file named "datatypes.py". The code contains a multi-line comment starting with a triple quote on line 3 and ending with another triple quote on line 6. Two red arrows point to these triple quotes. The status bar at the bottom shows the terminal output and some file navigation information.

If you want to make a
really large comment.

Surround your code with
““ at the top of the code
and ”” at the bottom

variables + data types



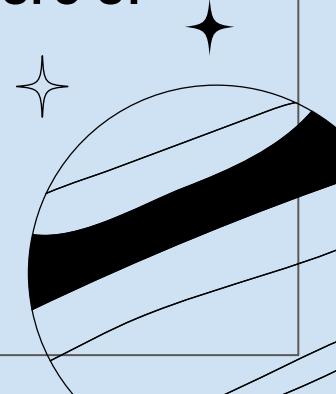


DATA TYPES overview

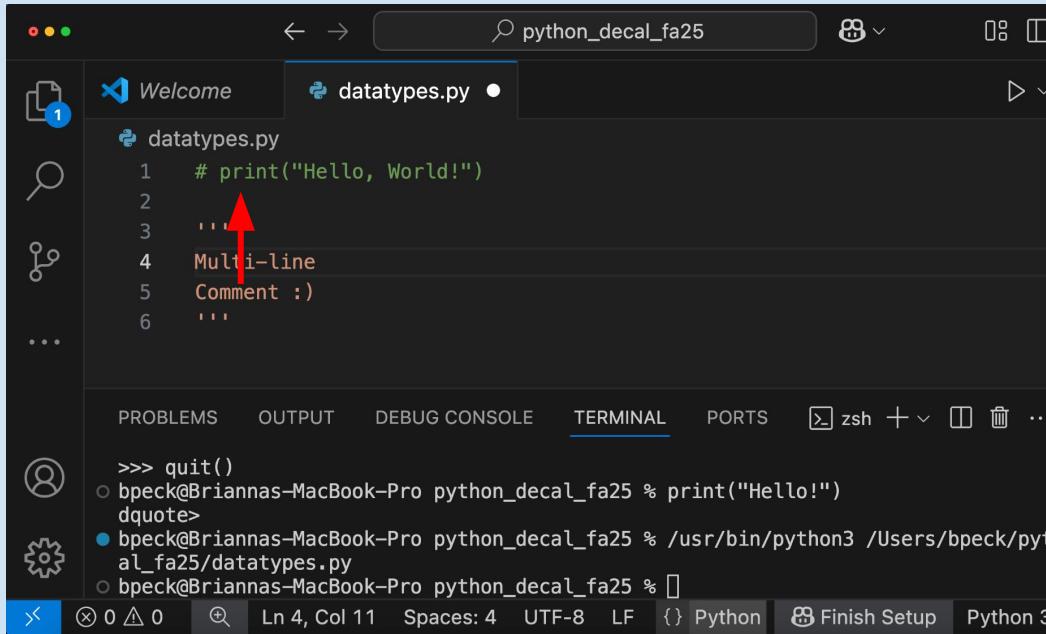
In the demo, we will cover three different data types today:

- Strings = words or text
- Integers = whole numbers
- Floats = fractional numbers or numbers with decimals

There are a ton of data types in Python, but you discover more of them in this week's homework.



PYTHON SYNTAX



A screenshot of a dark-themed code editor, likely VS Code, showing a file named `datatype.py`. The code contains a multi-line comment starting with `'''` and ending with `'''`. A red arrow points to the start of the first line of this comment. The code also includes a single-line print statement at the top.

```
# print("Hello, World!")  
'''  
Multi-line  
Comment :)  
'''
```

The terminal below shows the output of running the script:

```
>>> quit()  
bpeck@Briannas-MacBook-Pro python_decal_fa25 % print("Hello!")  
dquote>  
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/pyt  
al_fa25/datatypes.py  
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

At the bottom, status bar details include: Ln 4, Col 11, Spaces: 4, UTF-8, LF, Python, Finish Setup, Python 3.

All you have learned
about Python so far is just
the print() function

Function = reusable code

Calling the built-in print()
function tells Python to
output the phrase you
wrote



creating a variable



A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows the file name "python_decal_fa25". The left sidebar has icons for File, Welcome, and the current file "datatypes.py". The main editor area displays the following Python code:

```
datatypes.py > ...
5 Comment :)
6 ...
7
8 x = "Hello, World!"      Psst, this line is the code for LC2!
9 print(x)
```

The status bar at the bottom shows the terminal output:

```
>>> quit()
bpeck@Briannas-MacBook-Pro python_decal_fa25 % print("Hello!")
dquote>
● bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/pyt
al_fa25/datatypes.py
○ bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

The status bar also indicates the cursor is at Ln 9, Col 9, with 4 Spaces, in UTF-8 LF mode, using Python 3.

To create a variable in python, use the equals = symbol

Variable = holds information in Python

Type:

x = “Hello, World!”

print(x)





creating a variable



The screenshot shows a dark-themed instance of Visual Studio Code. In the top bar, the file 'datatypes.py' is open. The code in the editor is:

```
5  Comment :)
6  ...
7
8  x = "Hello, World!"
9  print(x)
```

The line 'print(x)' is highlighted with a yellow selection. A red box highlights the right-pointing arrow icon in the top right corner of the editor area. Below the editor is a terminal window titled 'Terminal (^)'. The terminal output shows:

```
>>> quit()
bpeck@Briannas-MacBook-Pro python_decal_fa25 % print("Hello!")
dquote>
● bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py
○ bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

The status bar at the bottom indicates the cursor is at 'Ln 9, Col 9'.

Overview: we assigned the variable x the words: Hello, World!

Then we are telling Python to print our variable

Click the arrow to run



creating a variable



```
python_decal_fa25
```

```
datatypes.py
```

```
datatype.py > ...
```

```
5 Comment :)
```

```
6 '''
```

```
7
```

```
8 x = "Hello, World!"
```

```
9 print(x)
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh + ~
```

```
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py
```

```
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py
```

```
Hello, World!
```

```
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

```
Ln 9, Col 9 Spaces: 4 UTF-8 LF {} Python Finish Setup Python 3.
```

Overview: We assigned the variable x to the words: Hello, World!

Then we are telling Python to print our variable

Click the red arrow to run

DATA TYPE OF a variable

```
python_decal_fa25
Welcome datatypes.py
datatypes.py > ...
1
2
3
4
5
6
7
8 x = "Hello, World!"
9 print(x)
10 print(type(x))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh + ~ ...

bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py
Hello, World!
bpeck@Briannas-MacBook-Pro python_decal_fa25 %

Ln 10, Col 15 Spaces: 4 UTF-8 LF {} Python Finish Setup Python 3.6

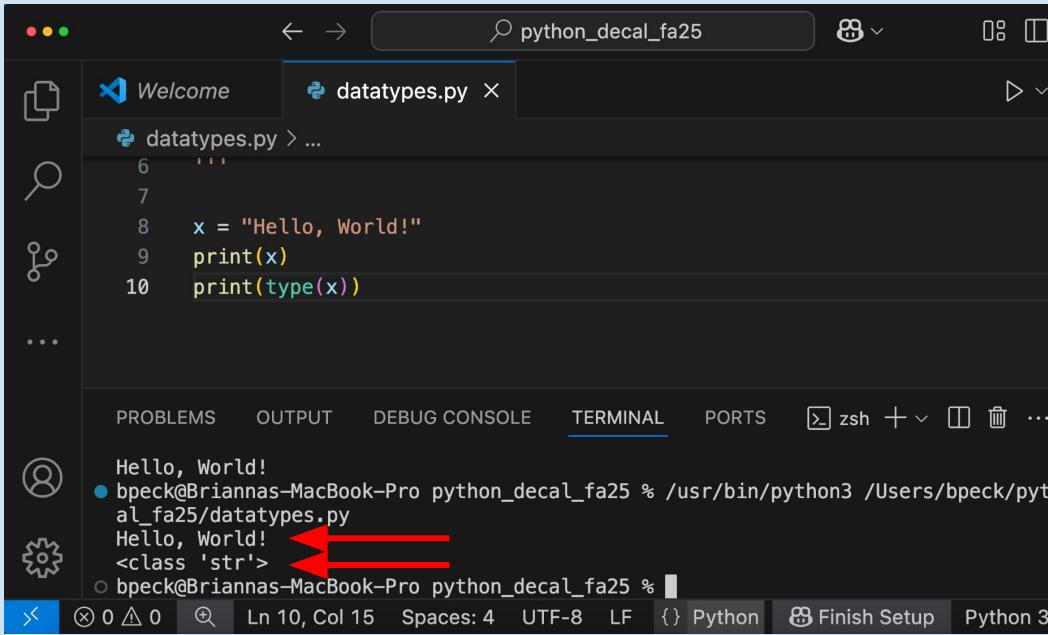
To check the data type:
print(type(x))

Now we have covered two built-in functions:

- **print()** = shows output
- **type()** = shows type

Click the arrow in the box

DATA TYPE: STRING



```
python_decal_fa25
Welcome datatypes.py > ...
datatypes.py > ...
1  ''
2  '
3  '
4  '
5  '
6  ''
7  '
8  x = "Hello, World!"
9  print(x)
10 print(type(x))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL zsh + ...
TERMINAL
Hello, World!
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/pyt
al_fa25/datatypes.py
Hello, World!
<class 'str'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

The terminal will now output two different lines

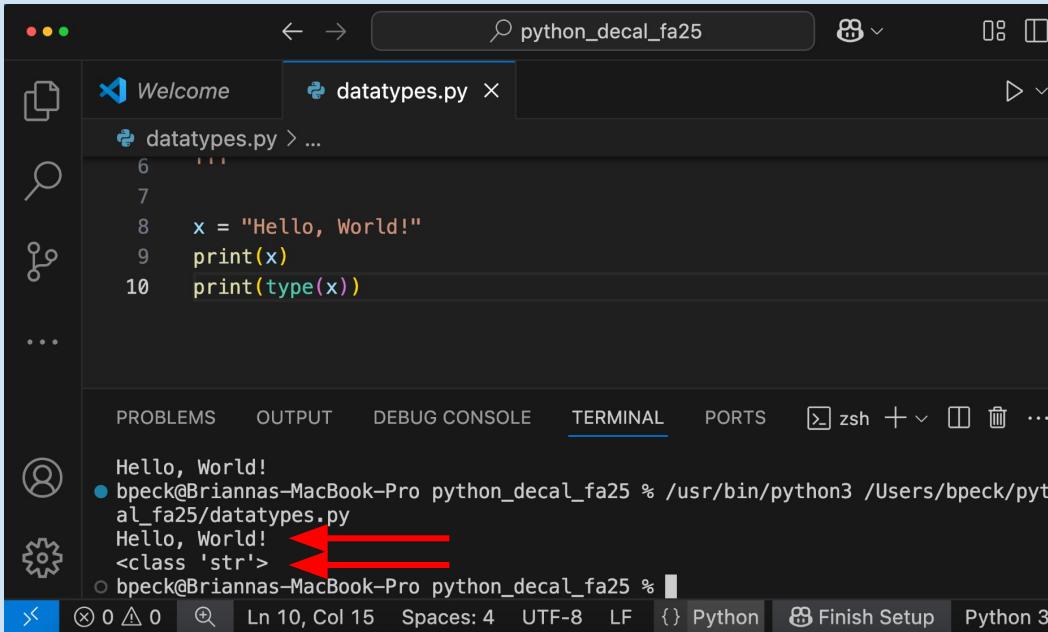
1st: "Hello, World!"

- Original statement

2nd: <class 'str'>

- str, stands for string
- Ignore the word "class" for now

DATA TYPE: STRING



A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows the search term "python_decal_fa25". The left sidebar has icons for Welcome, datatypes.py (the current file), and other workspace files. The main editor area contains the following Python code:

```
1  """
2
3  x = "Hello, World!"
4  print(x)
5  print(type(x))
```

The terminal at the bottom shows the output of running the script:

```
Hello, World!
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/pyt
al_fa25/datatypes.py
Hello, World! <class 'str'>
```

Two red arrows point to the output line "Hello, World! <class 'str'>" in the terminal window.

String = a data type that holds text. That's it!

You can't do a lot of operations on strings, but they are very helpful for figuring out what your program is doing

use DIFFerent NAMES

```
python_decal_fa25
datatypes.py
8 x = "Hello, World!" ←
9 print(x)
10 print(type(x))
11
12 x = 5 ←

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL zsh ...
TERMINAL
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py
Hello, World!
<class 'str'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

Note: If you reuse the same variable in Python. It will override the previous entry!

Notice how we have:

- **x = “Hello, World!”**
- **x = 5**

What happens if we call print(x)? Discuss.



use DIFFerent names



```
x = "Hello, World!"  
print(x)  
print(type(x))  
  
x = 5  
print(x)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh + ...

```
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py  
Hello, World!  
<class 'str'>  
5
```

Ln 13, Col 9 Spaces: 4 UTF-8 LF { } Python Finish Setup Python 3.

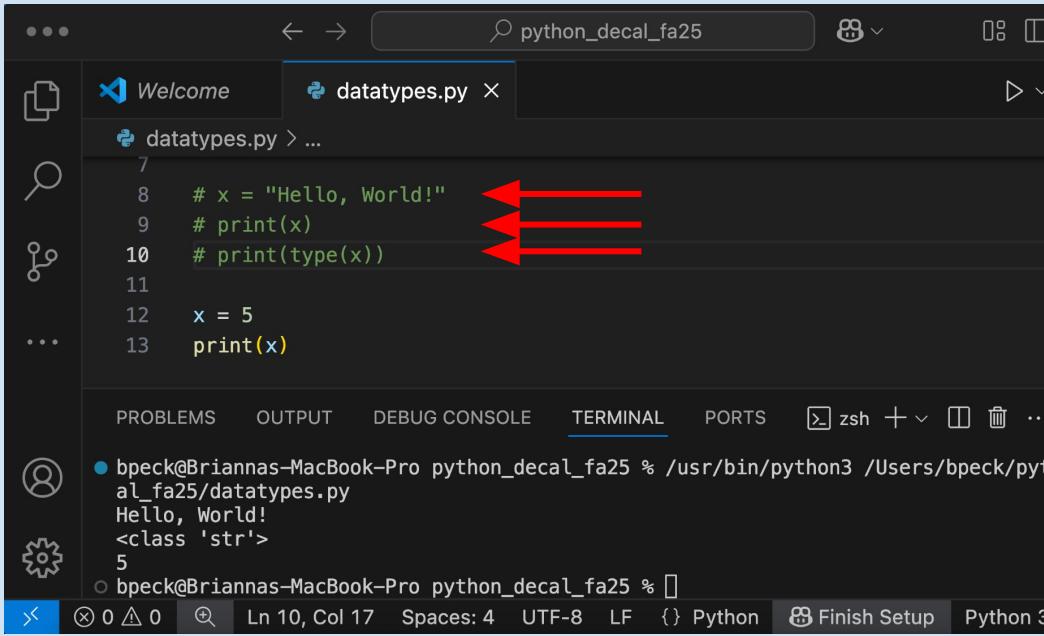
Type: print(x)

Instead of returning
“Hello, World!” like the
first x, Python now
returns 5!

Just keep this in mind, it's
good to use different
variable names



comment out code



```
# x = "Hello, World!"  
# print(x)  
# print(type(x))  
  
x = 5  
print(x)
```

The screenshot shows a dark-themed version of the Visual Studio Code interface. In the center, there's a code editor window with a Python file named 'datatypes.py'. The code contains three lines that print 'Hello, World!' and its type. Three red arrows highlight the first two lines of code. Below the code editor is a terminal window showing the output of running the script: 'Hello, World!' followed by '<class 'str'>'. At the bottom, the status bar displays file statistics like 'Ln 10, Col 17' and encoding information like 'UTF-8 LF'.

To avoid multiple outputs in the terminal as we move on... comment out the top chunk of code

A short cut is highlight the text you want to comment, then call:
Cmd + Q

DATA TYPES: INTEGER

```
# x = "Hello, World!"  
# print(x)  
# print(type(x))  
  
a = 5 # renaming to a to avoid confusion  
print(a)  
print(type(a))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh + ~

```
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py  
Hello, World!  
<class 'str'>  
5
```

x ⑧ 0 ▲ 0 + Ln 14, Col 15 Spaces: 4 UTF-8 LF {} Python Finish Setup Python 3.

To avoid confusion, let's rename the second "x" to a

Next let's print a and print what type of variable it is

Click the arrow in the box

DATA TYPES: INTEGER

```
# x = "Hello, World!"  
# print(x)  
# print(type(x))  
  
a = 5 # renaming to a to avoid confusion  
print(a)  
print(type(a))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh + ...

```
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py  
Hello, World!  
<class 'str'>  
5
```

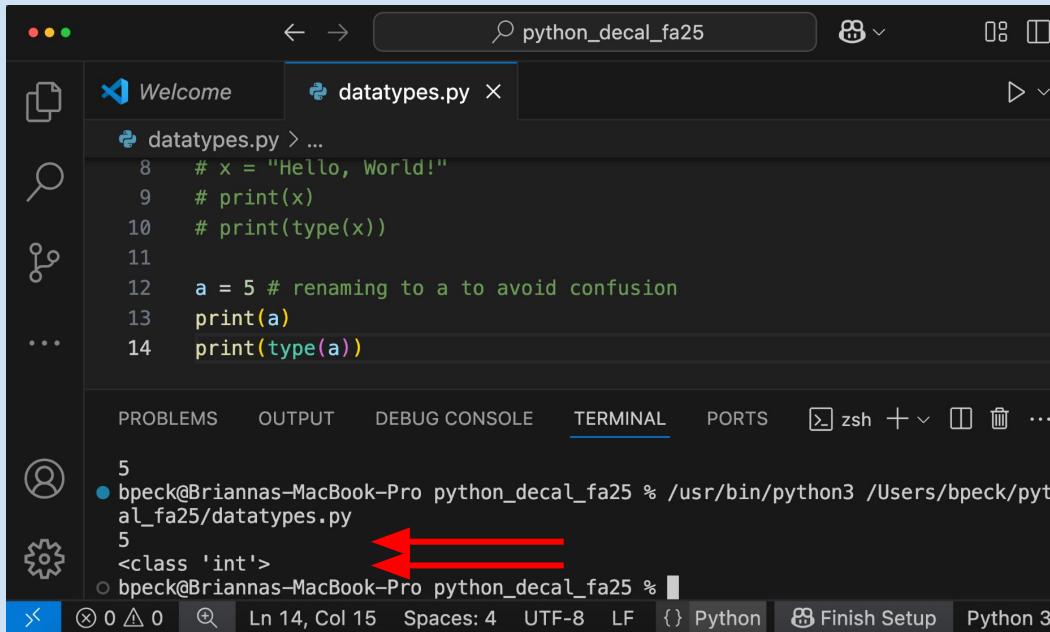
x 0 ▲ 0 + Ln 14, Col 15 Spaces: 4 UTF-8 LF {} Python Finish Setup Python 3.

To avoid confusion, let's rename the second "x" to a, we can add a comment

Next let's print a and print what type of variable it is

Click the arrow in the box

DATA TYPES: INTEGER



A screenshot of a code editor showing a file named `datatype.py`. The code contains the following Python script:

```
# x = "Hello, World!"  
# print(x)  
# print(type(x))  
  
a = 5 # renaming to a to avoid confusion  
print(a)  
print(type(a))
```

The terminal tab at the bottom shows the output of running the script:

```
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/python_decal_fa25/datatypes.py  
5  
<class 'int'>
```

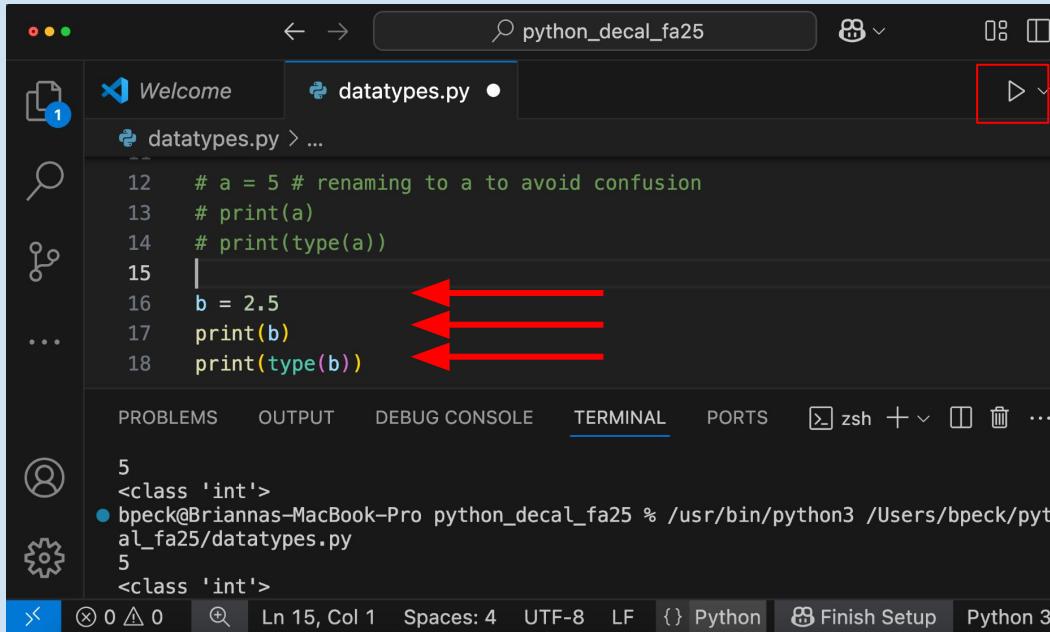
A red arrow points to the word `int` in the terminal output.

On the terminal, we see the “5” output again

But now we see a new data type: int

int = integer, a whole number

DATA TYPES: FLOAT



```
# a = 5 # renaming to a to avoid confusion
# print(a)
# print(type(a))
|
b = 2.5
print(b)
print(type(b))
```

First comment out the code about a.

Then type:

- b = 2.5
- print(b)
- print(type(b))

Click the arrow in the box

DATA TYPES (CONT.)

```
python_decal_fa25
datatypes.py > ...
12 # a = 5 # renaming to a to avoid confusion
13 # print(a)
14 # print(type(a))
15
16 b = 2.5
17 print(b)
18 print(type(b))

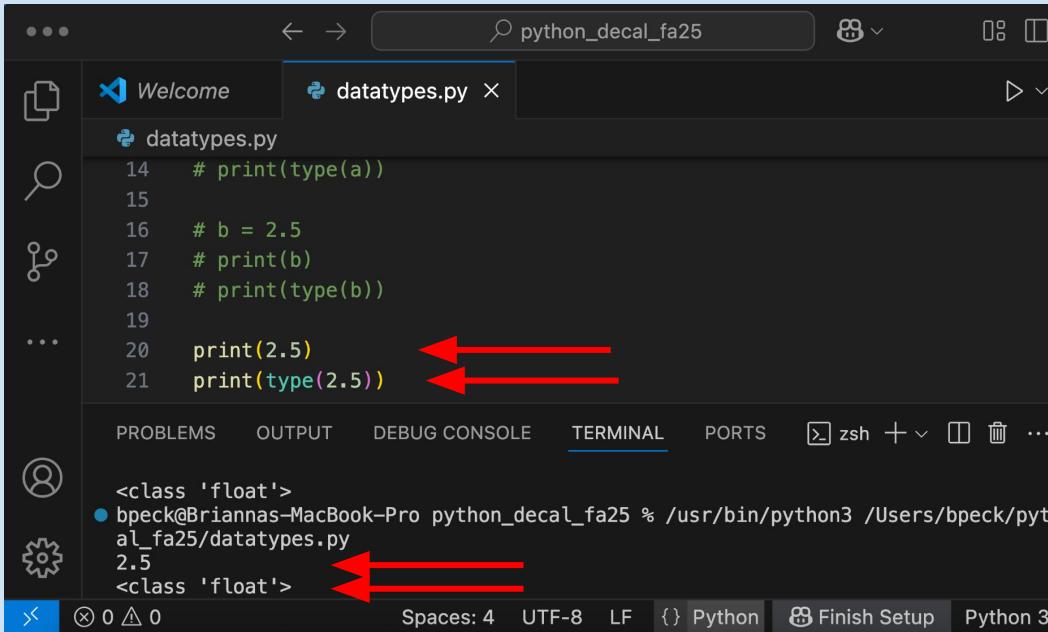
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL zsh + ... zsh
● bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/pyt
al_fa25/datatypes.py
2.5
<class 'float'>
○ bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

The terminal outputs b as 2.5, which is to be expected

But now we get:
<class 'float'>

float = a fractional
number, or a number with
a decimal

DATA TYPES (CONT.)



```
python_decal_fa25
datatypes.py
14 # print(type(a))
15
16 # b = 2.5
17 # print(b)
18 # print(type(b))
19
20 print(2.5)
21 print(type(2.5))

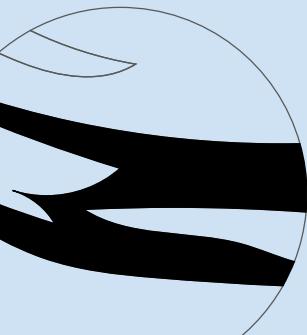
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh + □ □ ... zsh + □ □ ...
<class 'float'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/pyt
al_fa25/datatypes.py
2.5
<class 'float'>

Spaces: 4 UTF-8 LF {} Python Finish Setup Python 3.7
```

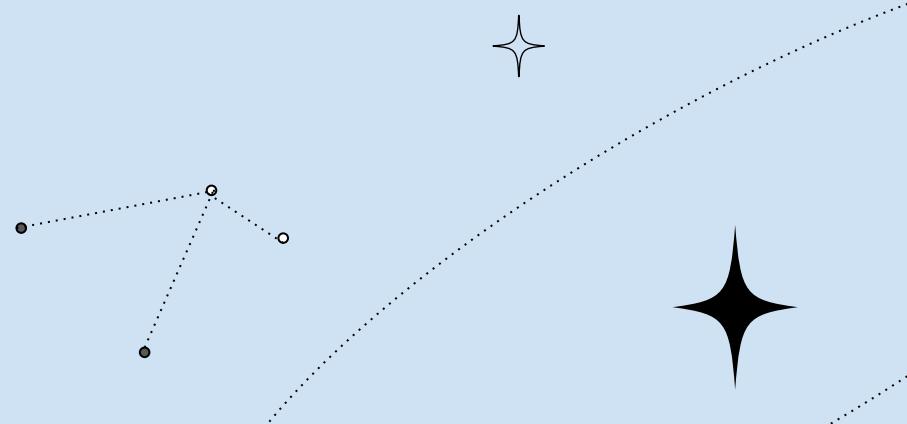
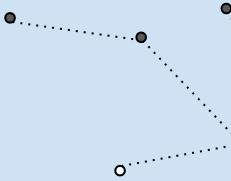
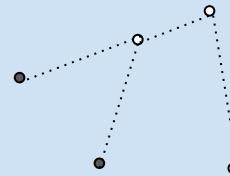
Note: You don't have to use a variable to use the built-in type() function

Just like print(), you can directly type into your type() function

Click the arrow to run



Let's Practice





PRACTICE QUESTION



If there is time, let's answer these questions:

Close those laptops! With a partner, discuss the following.

- 1) How would you write a script to assign a variable to:
 - [0, 1, 2]
- 2) How would you print that variable?
- 3) How would you determine the data type of that variable?
- 4) How would you run the script on VS Code?



PRACTICE QUESTION ANSWER

```
python_decal_fa25
```

```
datatypes.py
```

```
23  c = [0, 1, 2]
24  print(c)
25  print(type(c))
```

```
bpeck@Briannas-MacBook-Pro python_decal_fa25 % /usr/bin/python3 /Users/bpeck/pyt
al_fa25/datatypes.py
[0, 1, 2]
<class 'list'>
```

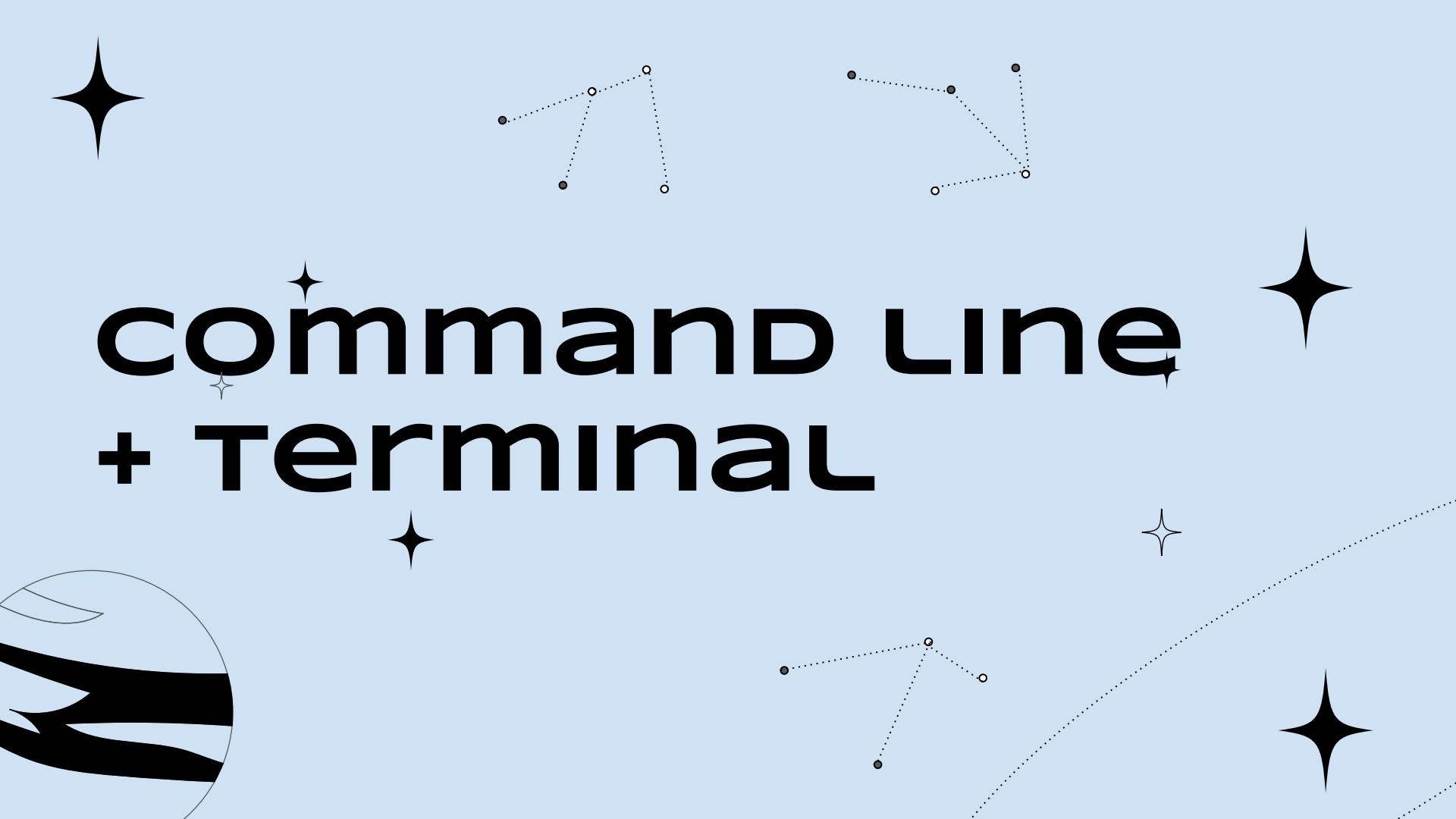
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh + ...

x 0 ▲ 0 + Ln 25, Col 15 Spaces: 4 UTF-8 LF {} Python Finish Setup Python 3.

To the left:

Answers to (1), (2), (3)

4) Click the arrow to run the script in the VS Code terminal



command line + terminal



TERMINAL BACKGROUND



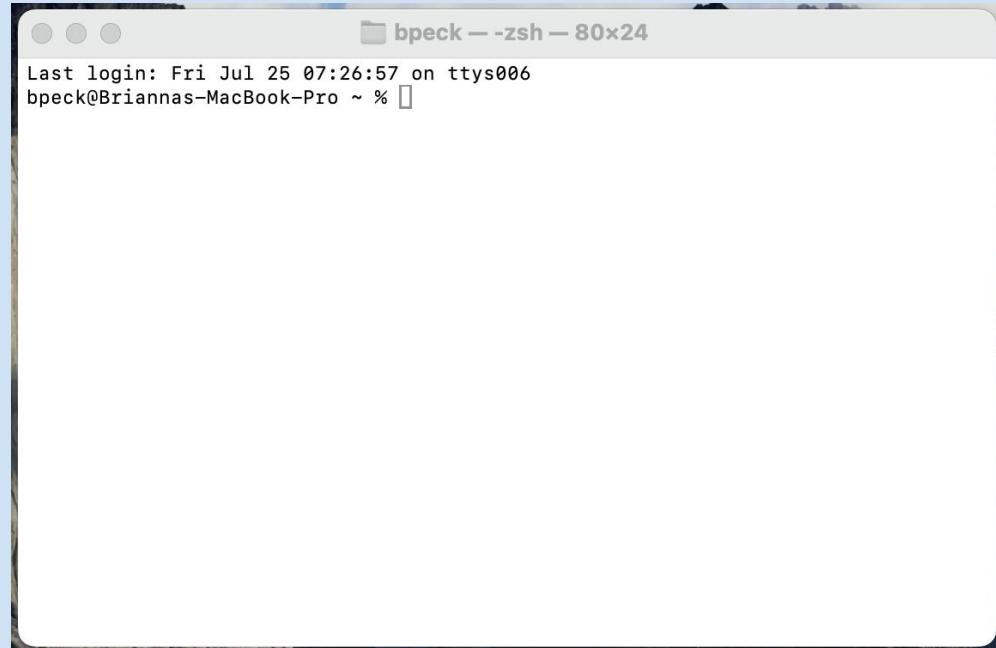
We already started working with the terminal on VS Code

Now we will work with the actual application on your computer

Windows users: Make sure you are on Git Bash!!!

- Not PowerShell





OPEN THE TERMINAL

Moving away from VS Code,
let's work directly with the
terminal

Search for Terminal on your
computer and open the
application

This is what mine looks like



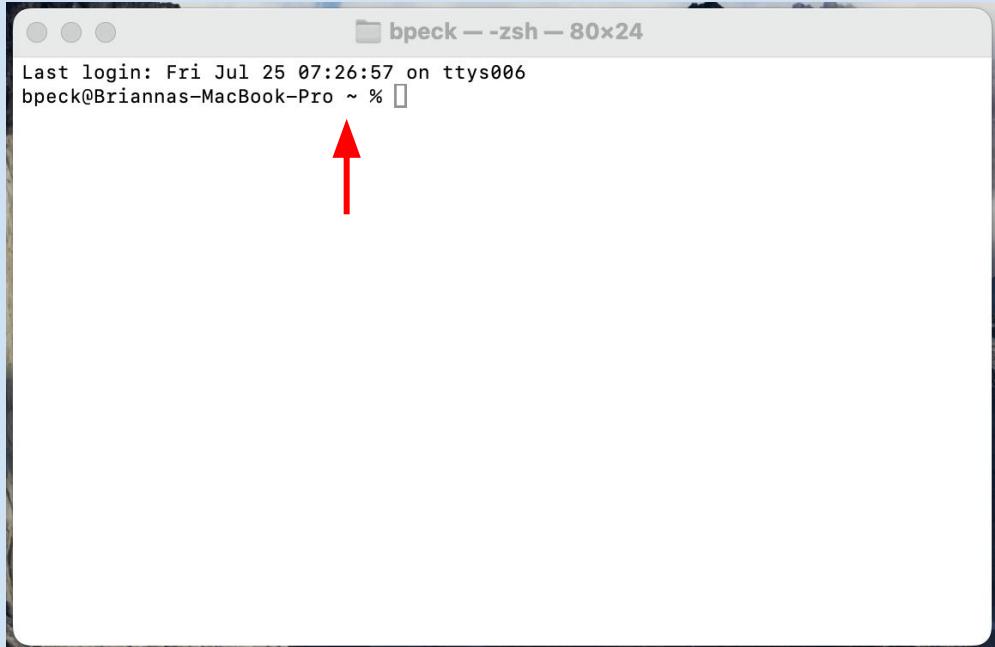
READY FOR A COMMAND

Last login: Fri Jul 25 07:26:57 on ttys006
bpeck@Briannas-MacBook-Pro ~ %



Just like with VS Code, we have the percentage % symbol which tell us that our command line is awaiting it's next command from us

HOME = DEFAULT DIRECTORY

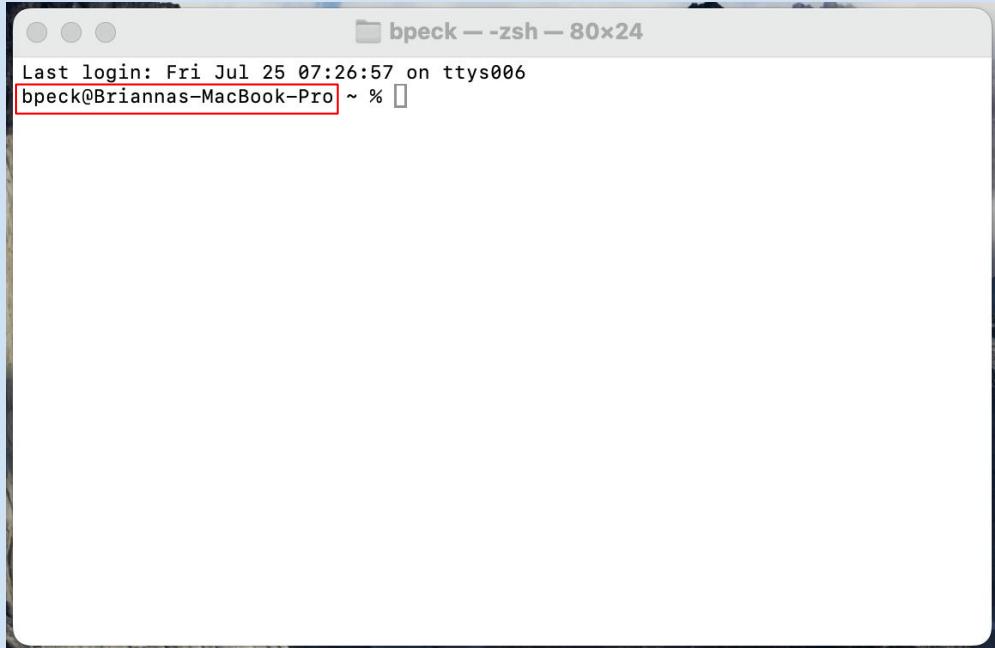


```
Last login: Fri Jul 25 07:26:57 on ttys006
bpeck@Briannas-MacBook-Pro ~ %
```

The tilde ~ symbol means that we are in our home directory

Home directory = default location when you first open up your terminal, designated by the ~ symbol

HOME = DEFAULT DIRECTORY



```
Last login: Fri Jul 25 07:26:57 on ttys006  
bpeck@Briannas-MacBook-Pro ~ %
```

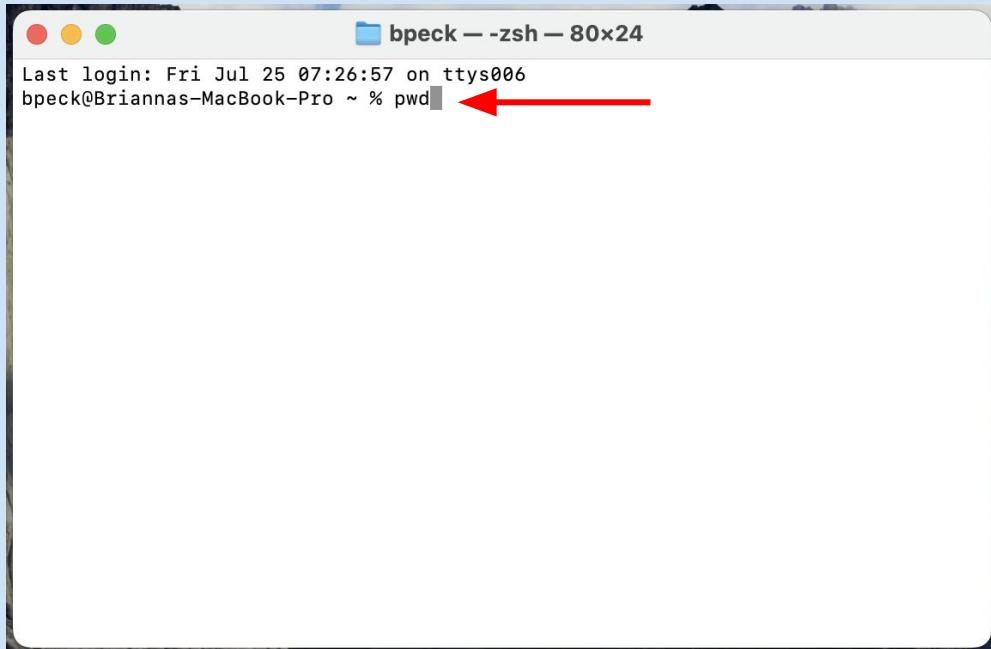
The stuff in the box tells you who is using the computer and what computer is being used

Users = bpeck

Computer:

- Briannas-MacBook-Pro
(it might also just be numbers = IP Address)

PWD = PRINT CWD



```
Last login: Fri Jul 25 07:26:57 on ttys006  
bpeck@Briannas-MacBook-Pro ~ % pwd
```

Type the command: pwd

This will tell your command line too: print the current working directory

current working directory = the folder you are currently in (we are in the “home” directory)



current working directory

```
Last login: Fri Jul 25 07:26:57 on ttys006  
bpeck@Briannas-MacBook-Pro ~ % pwd  
/Users/bpeck ←  
bpeck@Briannas-MacBook-Pro ~ %
```

My terminal tells me, that I
am currently in:
/Users/bpeck/

Each folder is separated by a
forward slash / symbol

So the top folder is Users,
which holds bpeck



THE ROOT DIRECTORY



```
Last login: Fri Jul 25 07:26:57 on ttys006  
bpeck@Briannas-MacBook-Pro ~ % pwd  
/Users/bpeck  
bpeck@Briannas-MacBook-Pro ~ %
```

The forward slash / at the very beginning of the line indicates the root directory

It's the very top folder of your entire computer, everything can be found from this folder

So: /Users/ = root directory



DIFFerences IN TERMS



```
Last login: Fri Jul 25 07:26:57 on ttys006
bpeck@Briannas-MacBook-Pro ~ % pwd
/Users/bpeck
bpeck@Briannas-MacBook-Pro ~ %
```

So:

/Users/ = root directory

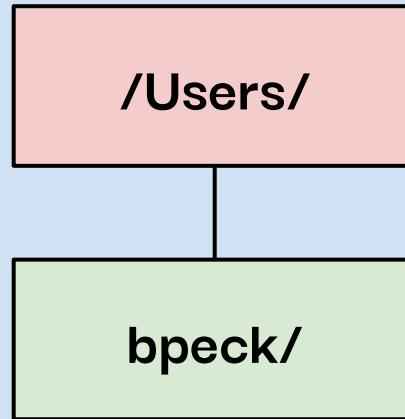
bpeck/ = a directory (folder)

**/Users/bpeck/ = home
directory and current
working directory**





QUICK NOTE: DIRECTORY TREES



Note: Directories can live in other directories

So bpeck/ is living in /Users/

Left: Is what is called a directory tree, we will work more with these later



DIFFerences IN TERMS



```
Last login: Fri Jul 25 07:26:57 on ttys006
bpeck@Briannas-MacBook-Pro ~ % pwd
/Users/bpeck
bpeck@Briannas-MacBook-Pro ~ %
```

This whole line:

/Users/bpeck

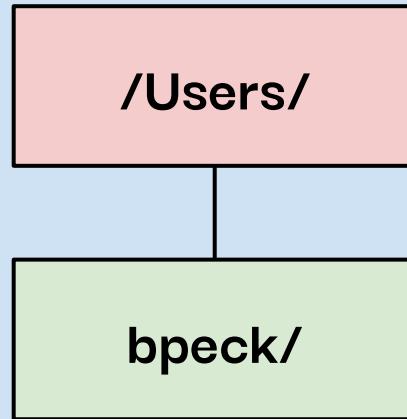
is called a file path

File path = an address on
your computer saying where
a folder or file is located





QUICK NOTE: DIRECTORY TREES



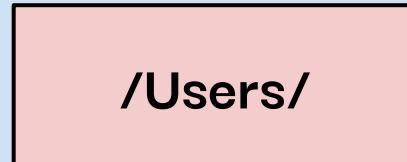
/Users/ is a different colors than
/bpeck/ because it is a special
directory

Red = root directory
Green = regular directory



Parent vs. CHILD DIRECTORIES

Parent



Child

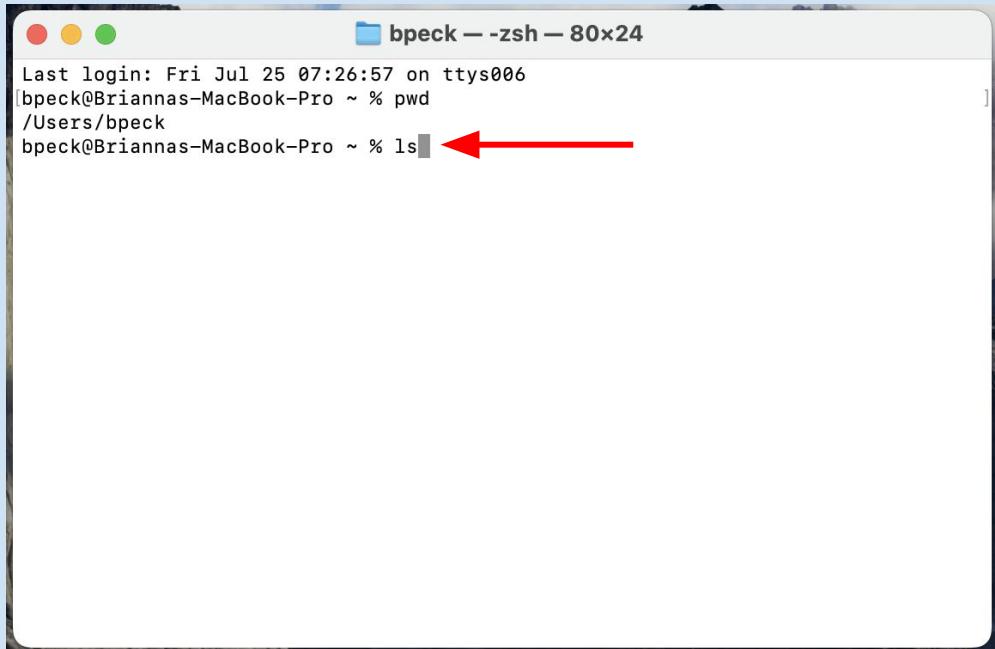
Directories are often referred to as parent and child directories

Parent = a folder that holds another folder

Child = the folder being held

So /Users/ is the parent and bpeck/ is the child

LS = LIST



A screenshot of a macOS terminal window titled "bpeck -- zsh -- 80x24". The window shows the following text:

```
Last login: Fri Jul 25 07:26:57 on ttys006  
bpeck@Briannas-MacBook-Pro ~ % pwd  
/Users/bpeck  
bpeck@Briannas-MacBook-Pro ~ % ls
```

A red arrow points to the "ls" command in the terminal.

Type: ls

This command tells your terminal to list the contents of your folder

It will give you all the files and folders*

*except hidden one

LS = LIST



```
/Users/bpeck
bpeck@Briannas-MacBook-Pro ~ % ls ←
239172_Keck_telescopes_f86d4028-8f20-4eb9-9e53-0466645d3409-prv.jpg
Applications
code
Creative Cloud Files bpeck114@berkeley.edu 44ad8df06b5bc65a895b4a91e41cd8250e626
410e71dfe095fe9f2bcd28d334
Creative Cloud Files Company Account University of California Berkeley bpeck114@berkeley.edu 42CD1E4C653DC6B90A495C16@845d757462d42564495fb4.e
Desktop
Documents
Downloads
get-pip.py
Library
Movies
Music
physics-111a-student
Pictures
Public
python_decal_fa25
python_decal_fa25_test
Test.py
ulab
bpeck@Briannas-MacBook-Pro ~ %
```

Each line after **ls**, shows a different file or folder

files = have extensions

- Like .py, .txt., .jpg

folders = have nothing



LOCATE NEW FOLDER



```
/Users/bpeck
bpeck@Briannas-MacBook-Pro ~ % ls
239172_Keck_telescopes_f86d4028-8f20-4eb9-9e53-0466645d3409-prv.jpg
Applications
code
Creative Cloud Files bpeck114@berkeley.edu 44ad8df06b5bc65a895b4a91e41cd8250e626
410e71dfe095fe9f2bcd28d334
Creative Cloud Files Company Account University of California Berkeley bpeck114@berkeley.edu 42CD1E4C653DC6B90A495C16@845d757462d42564495fb4.e
Desktop
Documents
Downloads
get-pip.py
Library
Movies
Music
physics-111a-student
Pictures
Public
python_decal_fa25 ←
python_decal_fa25_test
Test.py
ulab
bpeck@Briannas-MacBook-Pro ~ %
```

Search for the folder we made earlier called:
python_decal_sp26

It should be in your home directory

Windows Users: It may not be, if so, ask for help



CHANGE DIRECTORIES



```
/Users/bpeck
bpeck@Briannas-MacBook-Pro ~ % ls
239172_Keck_telescopes_f86d4028-8f20-4eb9-9e53-0466645d3409-prv.jpg
Applications
code
Creative Cloud Files bpeck114@berkeley.edu 44ad8df06b5bc65a895b4a91e41cd8250e626
410e71dfe095fe9f2bcd28d334
Creative Cloud Files Company Account University of California Berkeley bpeck114@berkeley.edu 42CD1E4C653DC6B90A495C16@845d757462d42564495fb4.e
Desktop
Documents
Downloads
get-pip.py
Library
Movies
Music
physics-111a-student
Pictures
Public
python_decal_fa25
python_decal_fa25_test
Test.py
ulab
bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25
```

Now type:

cd python_decal_sp26

**cd = change directories,
changes the folder you are
currently in**

**We are moving into a new
folder inside your home
directory**





CHANGE DIRECTORIES



```
bpeck@Briannas-MacBook-Pro ~ % ls  
239172_Keck_telescopes_f86d4028-8f20-4eb9-9e53-0466645d3409-prv.jpg  
Applications  
code  
Creative Cloud Files bpeck114@berkeley.edu 44ad8df06b5bc65a895b4a91e41cd8250e626  
410e71dfe095fe9f2bcd28d334  
Creative Cloud Files Company Account University of California Berkeley bpeck114@  
berkeley.edu 42CD1E4C653DC6B90A495C16@845d757462d42564495fb4.e  
Desktop  
Documents  
Downloads  
get-pip.py  
Library  
Movies  
Music  
physics-111a-student  
Pictures  
Public  
python_decal_fa25  
python_decal_fa25_test  
Test.py  
ulab  
bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25  
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

After you run the command,
your terminal might display
what folder you are in

If your computer acts like
mine, instead of ~ you now
see python_decal_sp26

If your terminal, doesn't
have this, no worries





CHANGE DIRECTORIES



```
bpeck@Briannas-MacBook-Pro ~ % ls  
239172_Keck_telescopes_f86d4028-8f20-4eb9-9e53-0466645d3409-prv.jpg  
Applications  
code  
Creative Cloud Files bpeck114@berkeley.edu 44ad8df06b5bc65a895b4a91e41cd8250e626  
410e71dfe095fe9f2bcd28d334  
Creative Cloud Files Company Account University of California Berkeley bpeck114@  
berkeley.edu 42CD1E4C653DC6B90A495C16@845d757462d42564495fb4.e  
Desktop  
Documents  
Downloads  
get-pip.py  
Library  
Movies  
Music  
physics-111a-student  
Pictures  
Public  
python_decal_fa25  
python_decal_fa25_test  
Test.py  
ulab  
[bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25  
bpeck@Briannas-MacBook-Pro python_decal_fa25 % pwd
```

To officially see what folder we are in now, type: **pwd** again





CHANGE DIRECTORIES



```
python_decal_fa25 -- zsh -- 80x24
Applications
code
Creative Cloud Files bpeck114@berkeley.edu 44ad8df06b5bc65a895b4a91e41cd8250e626
410e71dfa095fe9f2bcd28d334
Creative Cloud Files Company Account University of California Berkeley bpeck114@berkeley.edu 42CD1E4C653DC6B90A495C160845d757462d42564495fb4.e
Desktop
Documents
Downloads
get-pip.py
Library
Movies
Music
physics-111a-student
Pictures
Public
python_decal_fa25
python_decal_fa25_test
Test.py
ulab
bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25
bpeck@Briannas-MacBook-Pro python_decal_fa25 % pwd
/Users/bpeck/python_decal_fa25
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

We still see the original structure

root directory = /Users/
home directory = bpeck/

new directory =
/python_decal_sp26





CHANGE DIRECTORIES



```
python_decal_fa25 -- zsh -- 80x24
Applications
code
Creative Cloud Files bpeck114@berkeley.edu 44ad8df06b5bc65a895b4a91e41cd8250e626
410e71dfa095fe9f2bcdd28d334
Creative Cloud Files Company Account University of California Berkeley bpeck114@berkeley.edu 42CD1E4C653DC6B90A495C160845d757462d42564495fb4.e
Desktop
Documents
Downloads
get-pip.py
Library
Movies
Music
physics-111a-student
Pictures
Public
python_decal_fa25
python_decal_fa25_test
Test.py
ulab
bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25
bpeck@Briannas-MacBook-Pro python_decal_fa25 % pwd
/Users/bpeck/python_decal_fa25
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
```

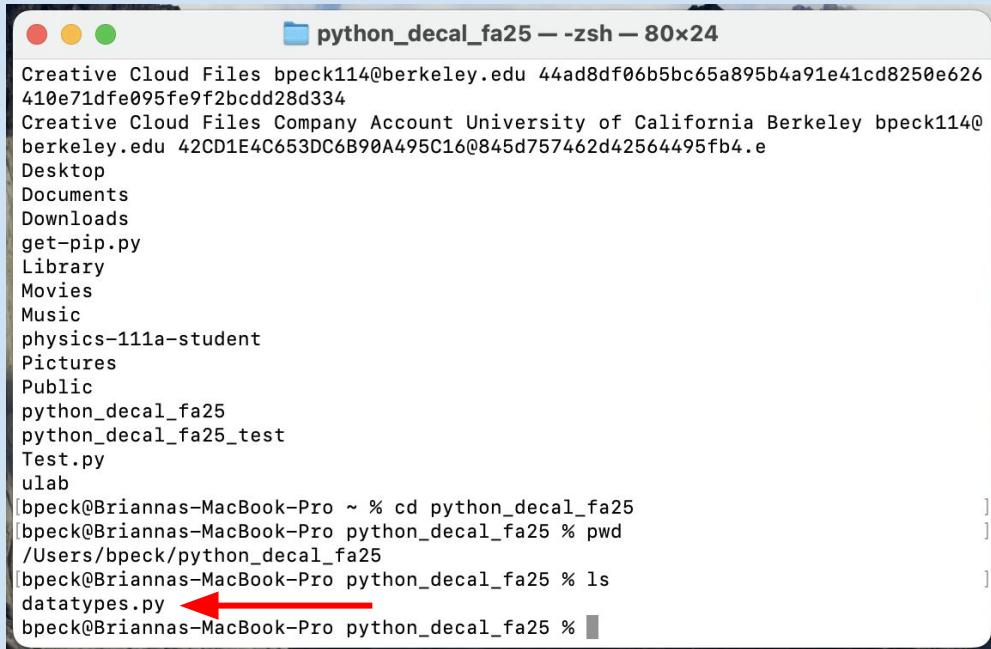
Type: ls

We are listing all the contents of the folder

Will it be empty? Discuss.



LIST CONTENTS



```
python_decal_fa25 -- zsh -- 80x24
Creative Cloud Files bpeck114@berkeley.edu 44ad8df06b5bc65a895b4a91e41cd8250e626
410e71dfe095fe9f2bcdd28d334
Creative Cloud Files Company Account University of California Berkeley bpeck114@berkeley.edu 42CD1E4C653DC6B90A495C160845d757462d42564495fb4.e
Desktop
Documents
Downloads
get-pip.py
Library
Movies
Music
physics-111a-student
Pictures
Public
python_decal_fa25
python_decal_fa25_test
Test.py
ulab
bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25
bpeck@Briannas-MacBook-Pro python_decal_fa25 % pwd
/Users/bpeck/python_decal_fa25
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatype.py ←
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

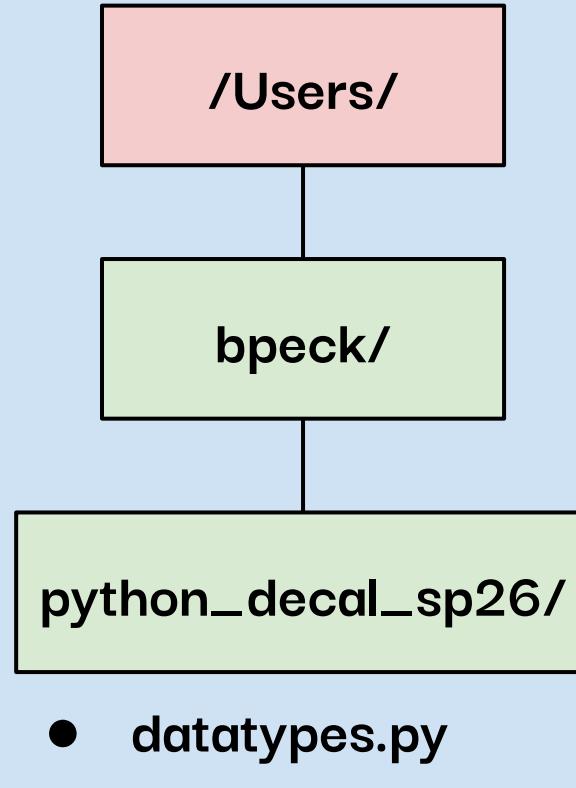
Nope! It's not empty.

It holds the file we made earlier in VS Code!

File: datatype.py



QUICK NOTE: DIRECTORY TREES



If we return to our directory tree again, we see a structure like the one to the left.

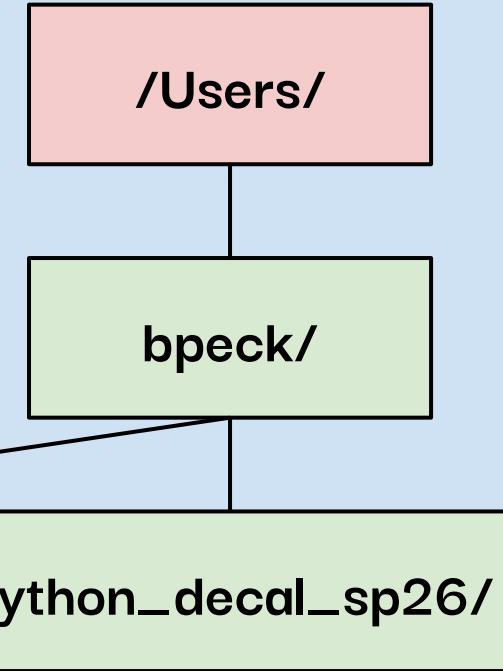
bpeck/ holds a lot of other files and folders, but we are not showing that currently

python_decal_sp26/ holds the file **datatypes.py**



QUICK NOTE: DIRECTORY TREES

Other
Stuff



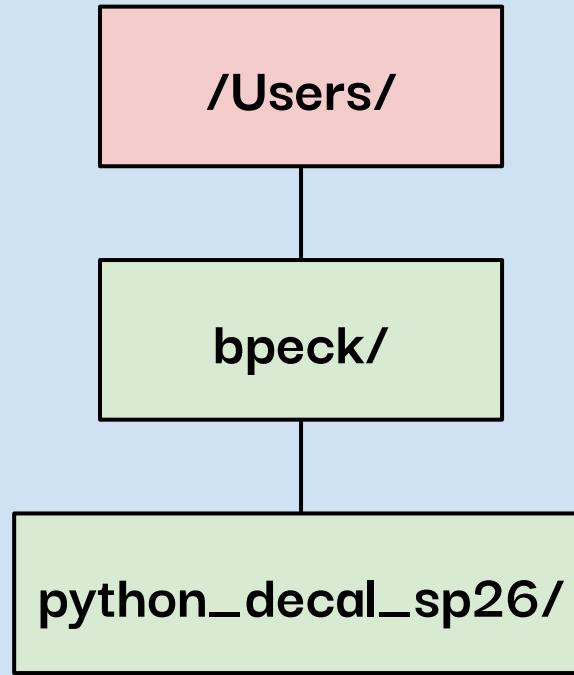
If we return to our directory tree again, we see a structure like the one to the left.

bpeck/ holds a lot of other files and folders, but we are not showing that currently

python_decal_sp26/ holds the file datatypes.py



QUICK NOTE: Parent/Child



- `datatype.py`

If we think about parent vs. child directories again:

/Users/ is a **parent** to bpeck/

bpeck/ is a **parent** to python.../

python.../ is a **child** to bpeck/

bpeck/ is a **child** to /Users/



concatenate a FILE



```
python_decal_fa25 -- zsh -- 80x24
Creative Cloud Files bpeck114@berkeley.edu 44ad8df06b5bc65a895b4a91e41cd8250e626
410e71dfe095fe9f2bcd28d334
Creative Cloud Files Company Account University of California Berkeley bpeck114@berkeley.edu 42CD1E4C653DC6B90A495C160845d757462d42564495fb4.e
Desktop
Documents
Downloads
get-pip.py
Library
Movies
Music
physics-111a-student
Pictures
Public
python_decal_fa25
python_decal_fa25_test
Test.py
ulab
bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25
bpeck@Briannas-MacBook-Pro python_decal_fa25 % pwd
/Users/bpeck/python_decal_fa25
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatype.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % cat datatype.py
```

Type: cat datatype.py

The cat command tells your computers to print out all the contents of a file

Cat = concatenate (😺)





concatenate a FILE



```
python_decal_fa25 --zsh-- 80x24
...
Multi-line
Comment :)
...
# x = "Hello, World!"
# print(x)
# print(type(x))

# a = 5 # renaming to a to avoid confusion
# print(a)
# print(type(a))

# b = 2.5
# print(b)
# print(type(b))

print(2.5)
print(type(2.5))

c = [0, 1, 2]
print(c)
print(type(c))
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

The terminal spit out every single line in our `datatypes.py` file



RUN A SCRIPT ON TERMINAL

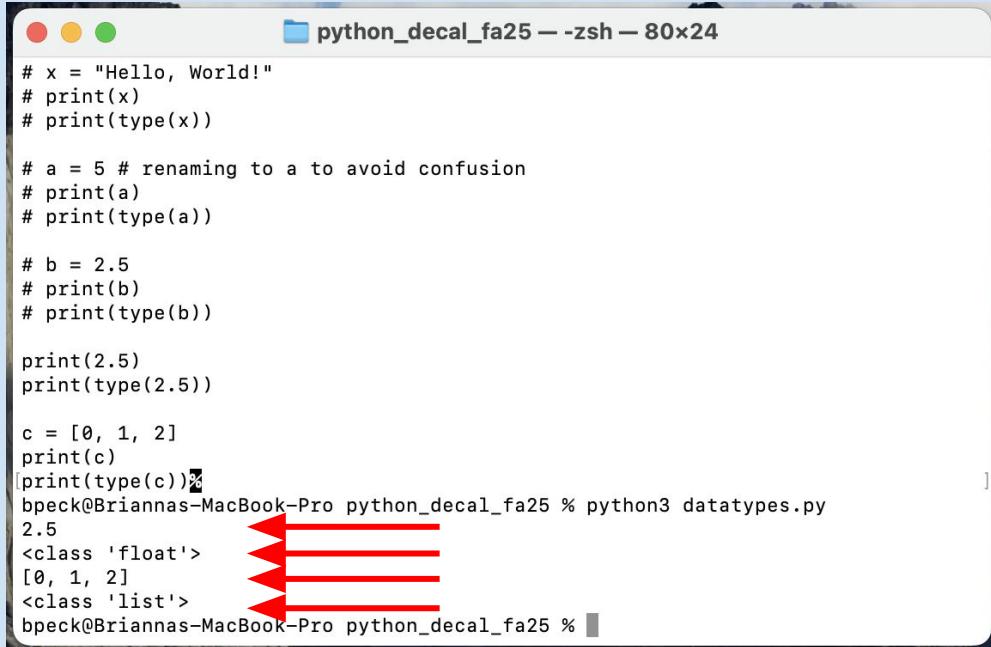
```
...  
Multi-line  
Comment :)  
...  
  
# x = "Hello, World!"  
# print(x)  
# print(type(x))  
  
# a = 5 # renaming to a to avoid confusion  
# print(a)  
# print(type(a))  
  
# b = 2.5  
# print(b)  
# print(type(b))  
  
print(2.5)  
print(type(2.5))  
  
c = [0, 1, 2]  
print(c)  
print(type(c))  
bbeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatype.py
```

To run a script on the terminal without VS Code, type the following:

[python3 datatype.py](#)

This tells your terminal to use python to run your script

RUN A SCRIPT ON TERMINAL



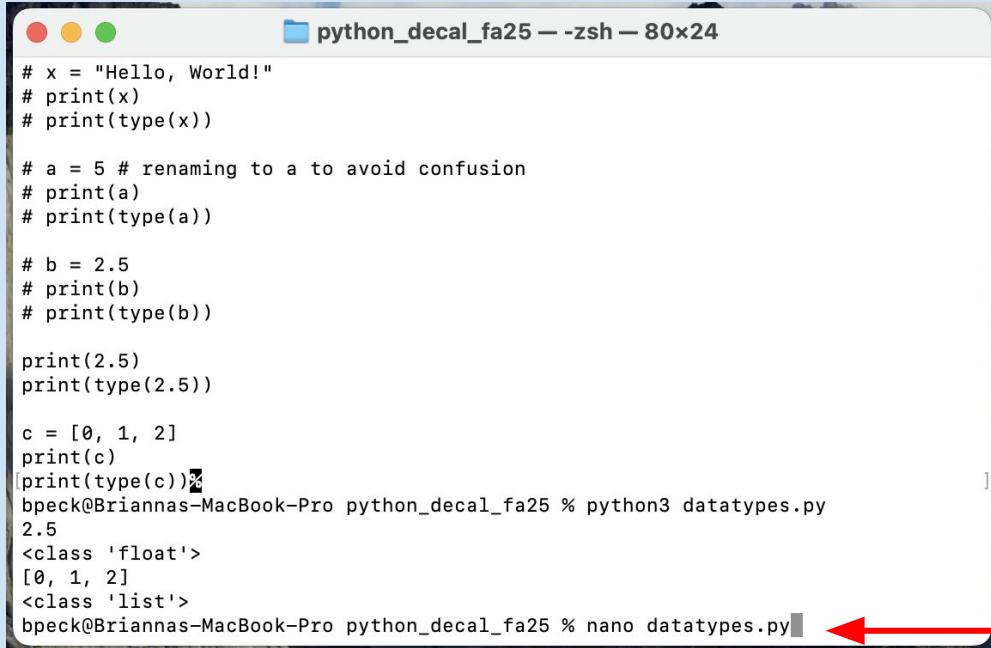
```
# x = "Hello, World!"  
# print(x)  
# print(type(x))  
  
# a = 5 # renaming to a to avoid confusion  
# print(a)  
# print(type(a))  
  
# b = 2.5  
# print(b)  
# print(type(b))  
  
print(2.5)  
print(type(2.5))  
  
c = [0, 1, 2]  
print(c)  
print(type(c))  
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py  
2.5  
<class 'float'>  
[0, 1, 2]  
<class 'list'>  
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

Three red arrows point from the printed output lines ('2.5', '<class \'float\'>', and '[0, 1, 2]') to the corresponding lines in the Python code above them.

You might have slightly different outputs if you didn't comment out as much as I did

But the terminal just ran our script in Python and returned the print statements

EDIT a FILE on THE TERMINAL



A screenshot of a terminal window titled "python_decal_fa25 --zsh-- 80x24". The window contains the following Python code:

```
# x = "Hello, World!"  
# print(x)  
# print(type(x))  
  
# a = 5 # renaming to a to avoid confusion  
# print(a)  
# print(type(a))  
  
# b = 2.5  
# print(b)  
# print(type(b))  
  
print(2.5)  
print(type(2.5))  
  
c = [0, 1, 2]  
print(c)  
print(type(c))  
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py  
2.5  
<class 'float'>  
[0, 1, 2]  
<class 'list'>  
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
```

A red arrow points to the command "nano datatypes.py" at the bottom of the terminal window.

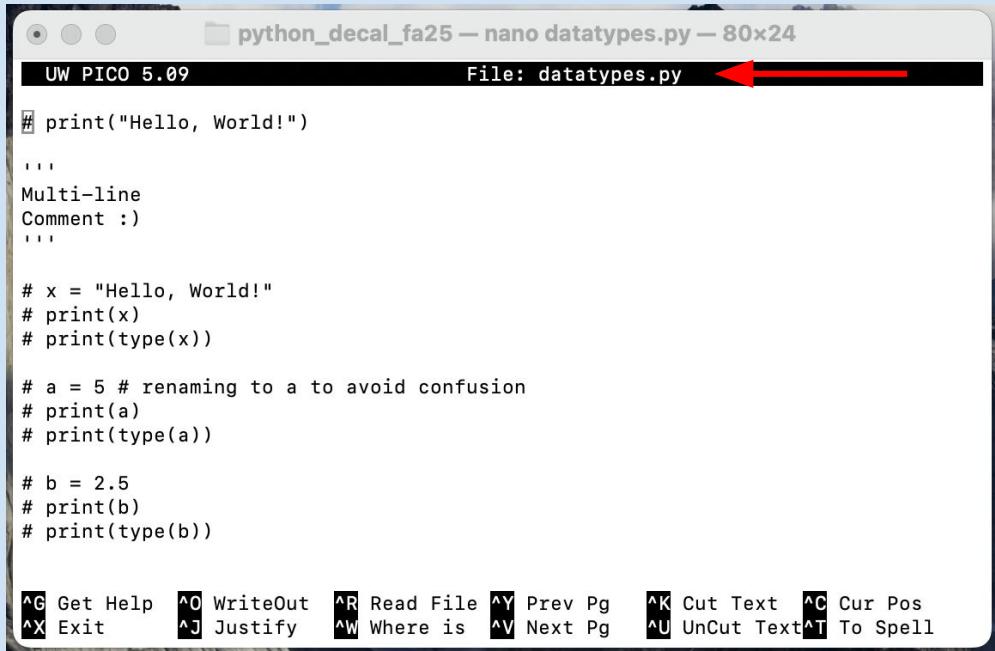
To edit your file on the command line:

Type: nano datatype.py

There are other commands that are similar to nano like:

- vim
- emacs
- micro

EDIT a FILE on THE TERMINAL



A screenshot of a terminal window titled "python_decal_fa25 — nano datatypes.py — 80x24". The window shows Python code for a file named "datatypes.py". A red arrow points to the title bar where the file name "datatypes.py" is displayed. The code itself is as follows:

```
# print("Hello, World!")

"""
Multi-line
Comment :)
"""

# x = "Hello, World!"
# print(x)
# print(type(x))

# a = 5 # renaming to a to avoid confusion
# print(a)
# print(type(a))

# b = 2.5
# print(b)
# print(type(b))
```

At the bottom of the terminal window, there is a menu of keyboard shortcuts:

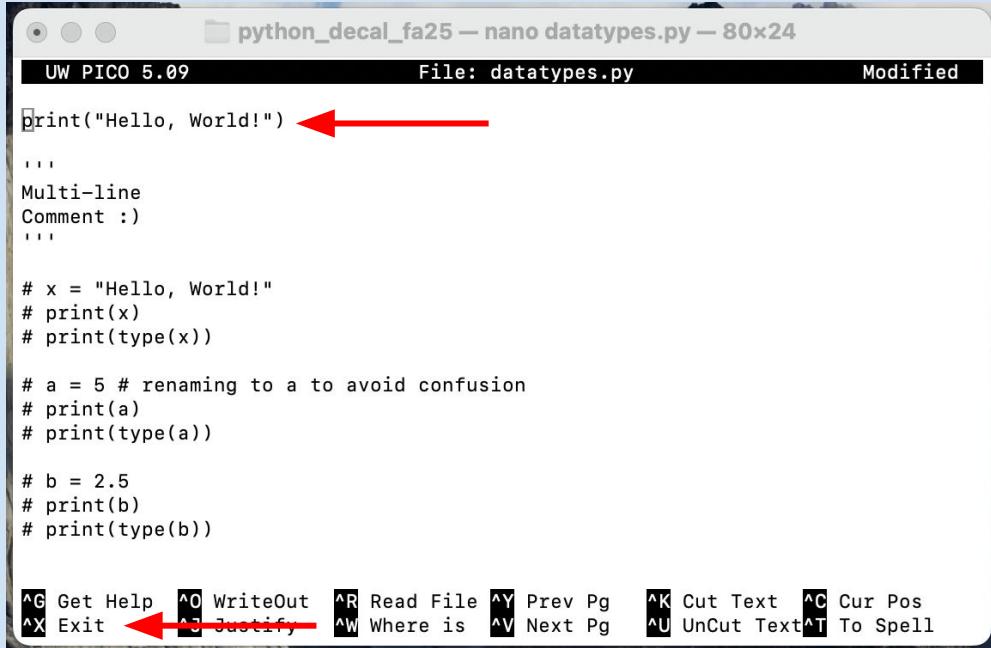
- ^G Get Help
- ^O WriteOut
- ^R Read File
- ^Y Prev Pg
- ^K Cut Text
- ^C Cur Pos
- ^X Exit
- ^J Justify
- ^W Where is
- ^V Next Pg
- ^U UnCut Text
- ^T To Spell

A new “window” appeared
but you are still on the
terminal

At the top is the name of the
file: datatypes.py

Below that, you have the file
contents

EDIT a FILE on THE TERMINAL



A screenshot of a terminal window titled "python_decal_fa25 — nano datatypes.py — 80x24". The window shows Python code. A red arrow points to the first line of code: `print("Hello, World!")`. Another red arrow points to the bottom-left corner of the terminal window, where the nano editor's command-line interface is visible.

```
UW PICO 5.09           File: datatypes.py      Modified  
print("Hello, World!")  
...  
Multi-line  
Comment :)  
...  
  
# x = "Hello, World!"  
# print(x)  
# print(type(x))  
  
# a = 5 # renaming to a to avoid confusion  
# print(a)  
# print(type(a))  
  
# b = 2.5  
# print(b)  
# print(type(b))  
  
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Pg  ^K Cut Text  ^C Cur Pos  
^X Exit  ^J Justify  ^W Where is  ^V Next Pg  ^U UnCut Text  ^T To Spell
```

Uncomment the very first
line in the file

Then press Ctrl + X

EDIT a FILE on THE TERMINAL

```
python_decal_fa25 — nano datatypes.py — 80x24
UW PICO 5.09          File: datatypes.py      Modified
print("Hello, World!")

...
Multi-line
Comment :)
...

# x = "Hello, World!"
# print(x)
# print(type(x))

# a = 5 # renaming to a to avoid confusion
# print(a)
# print(type(a))

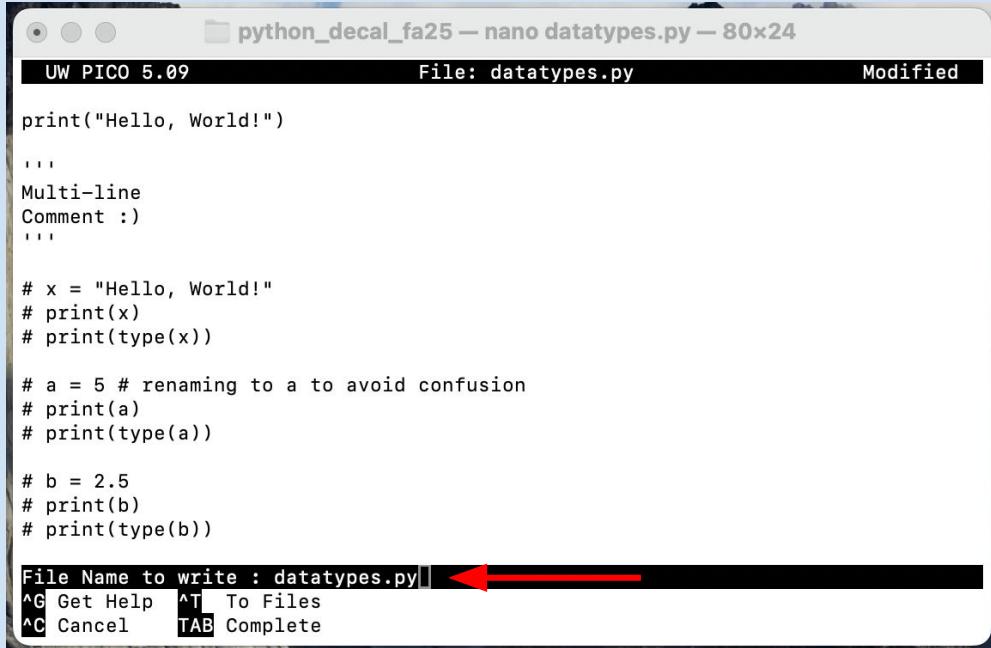
# b = 2.5
# print(b)
# print(type(b))

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ? [ ]
^C Cancel   Y Yes ←
N No
```

The nano window will prompt you will asking you if you want to save change

Press: Y to save your changes

EDIT a FILE ON THE TERMINAL



A screenshot of a terminal window titled "python_decal_fa25 — nano datatypes.py — 80x24". The window shows Python code:

```
UW PICO 5.09          File: datatypes.py      Modified
print("Hello, World!")

...
Multi-line
Comment :)
...

# x = "Hello, World!"
# print(x)
# print(type(x))

# a = 5 # renaming to a to avoid confusion
# print(a)
# print(type(a))

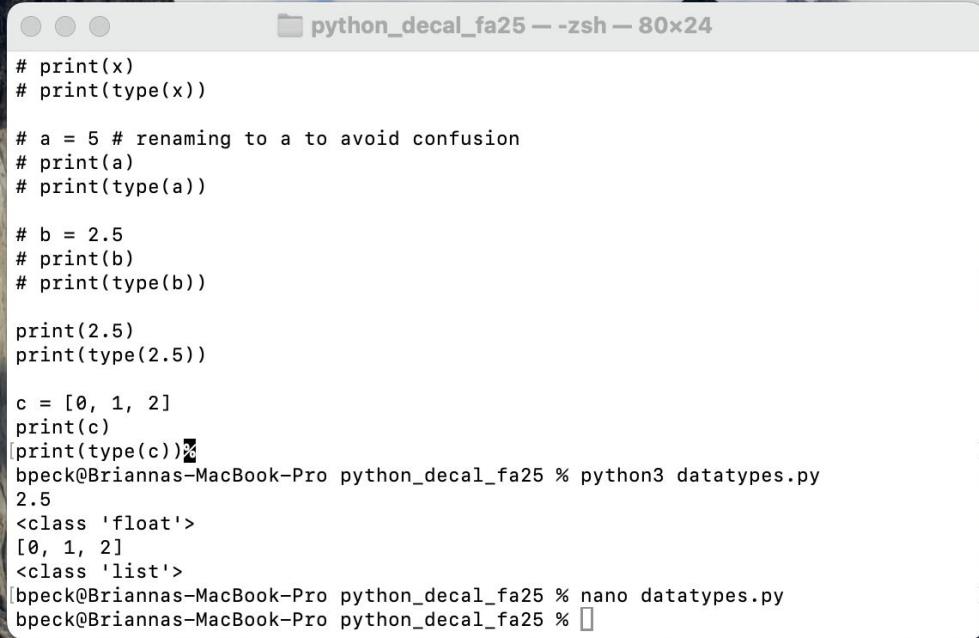
# b = 2.5
# print(b)
# print(type(b))

File Name to write : datatypes.py|
```

The bottom status bar shows keyboard shortcuts: ^G Get Help, ^T To Files, ^C Cancel, TAB Complete. A red arrow points to the command line input field where "datatypes.py" is typed.

Then it will ask what file you want to save your contents too

Just press: Return/Enter to not change the file name



A screenshot of a terminal window titled "python_decal_fa25 --zsh-- 80x24". The window contains the following Python code:

```
# print(x)
# print(type(x))

# a = 5 # renaming to a to avoid confusion
# print(a)
# print(type(a))

# b = 2.5
# print(b)
# print(type(b))

print(2.5)
print(type(2.5))

c = [0, 1, 2]
print(c)
print(type(c))
```

The terminal then shows the output of the code:

```
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
```

Finally, the user types "nano datatypes.py" and presses enter:

```
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ]
```

Once you are done, it will
dump you back to this
screen





RUN A SCRIPT ON THE TERMINAL

```
# print(x)
# print(type(x))

# a = 5 # renaming to a to avoid confusion
# print(a)
# print(type(a))

# b = 2.5
# print(b)
# print(type(b))

print(2.5)
print(type(2.5))

c = [0, 1, 2]
print(c)
print(type(c))
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
```

To run the script on your terminal again

Type:

python3 datatypes.py again





RUN A SCRIPT ON THE TERMINAL

The new “Hello, World!”
appeared on the terminal!

```
# b = 2.5
# print(b)
# print(type(b))

print(2.5)
print(type(2.5))

c = [0, 1, 2]
print(c)
print(type(c))%
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World! -----^
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```





MAKE A NEW FILE ON TERMINAL

```
# b = 2.5
# print(b)
# print(type(b))

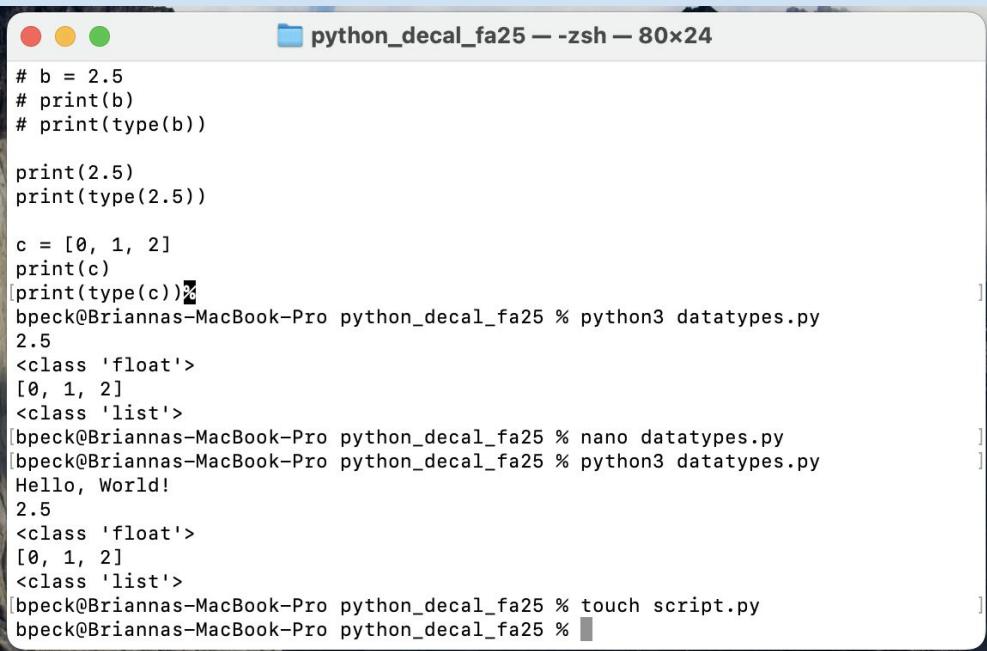
print(2.5)
print(type(2.5))

c = [0, 1, 2]
print(c)
print(type(c))%
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py%
```

To make a new file on the terminal type:

touch script.py

Note: You can also just call nano directly like:
nano script.py to open up an empty file.



A screenshot of a Mac OS X terminal window titled "python_decal_fa25 --zsh-- 80x24". The window contains the following Python code and its output:

```
# b = 2.5
# print(b)
# print(type(b))

print(2.5)
print(type(2.5))

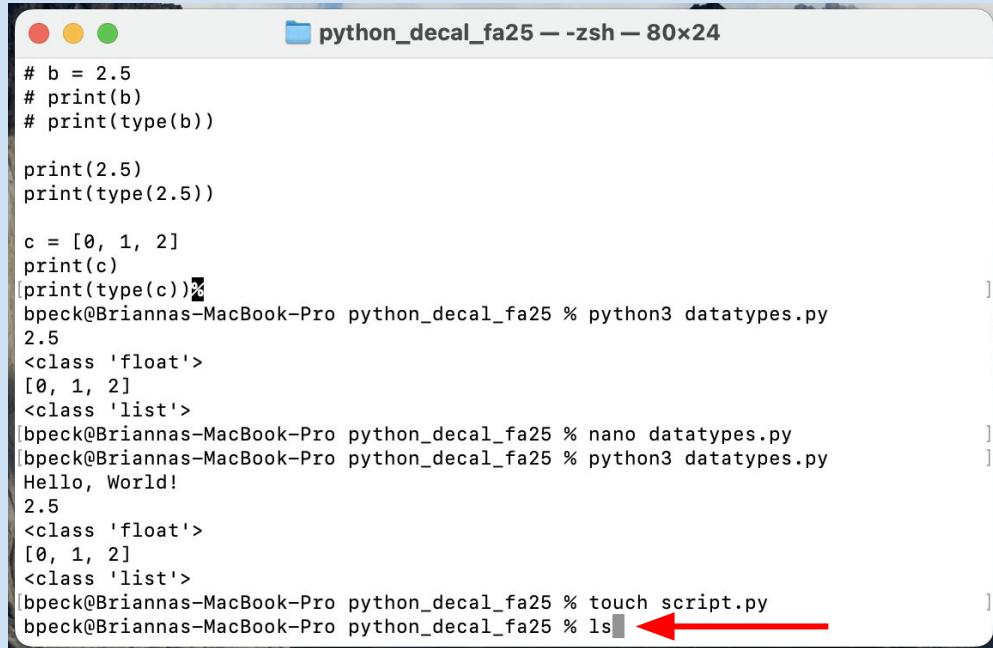
c = [0, 1, 2]
print(c)
print(type(c))
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

This is what your screen will look like after you call the command





MAKE A NEW FILE ON TERMINAL



A screenshot of a Mac OS X terminal window titled "python_decal_fa25 --zsh-- 80x24". The window shows the following command-line session:

```
# b = 2.5
# print(b)
# print(type(b))

print(2.5)
print(type(2.5))

c = [0, 1, 2]
print(c)
print(type(c))
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
```

A red arrow points to the "ls" command at the bottom of the terminal window.

If you type ls on the terminal, we will see two files now





MAKE A NEW FILE ON TERMINAL

```
# print(type(b))

print(2.5)
print(type(2.5))

c = [0, 1, 2]
print(c)
[print(type(c))]

bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>

bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>

bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatypes.py    script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

The terminal window shows the execution of a Python script named 'datatypes.py'. It prints the type of variables and lists. Then, it edits the same file using 'nano'. Finally, it creates a new file named 'script.py' using the 'touch' command.

If you type ls on the terminal, we will see two files now

datatypes.py

- which we just edited

script.py

- the new file we made



QUICK NOTE: DIRECTORY TREES

python_decal_sp26/

- datatypes.py
- script.py

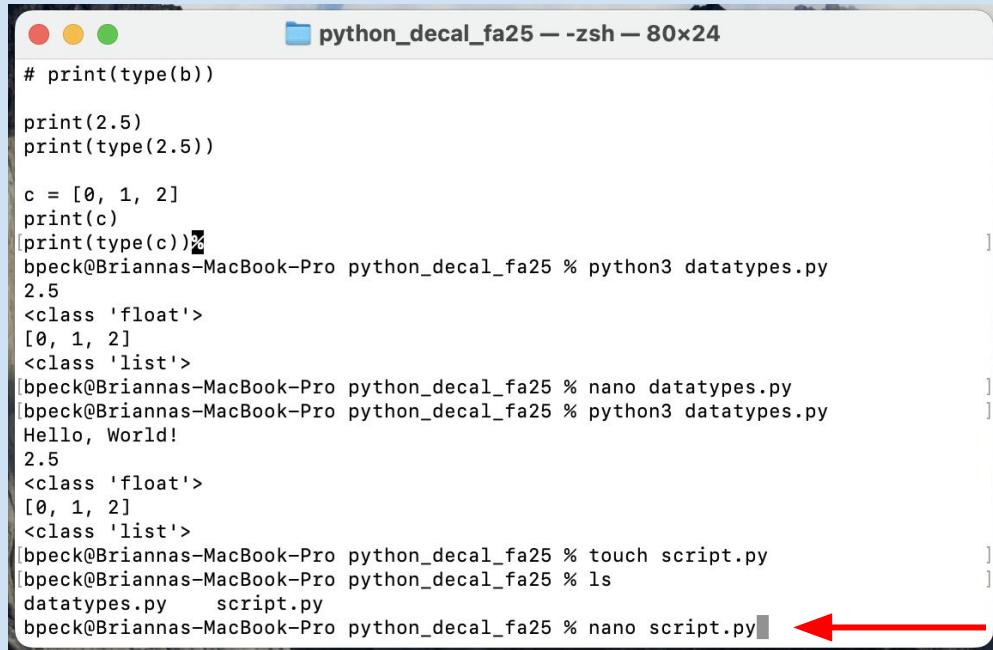
Let's get rid of /Users/ and bpeck/ in our directory tree because we are running out of space..

Now our tree looks like this:





MAKE A NEW FILE ON TERMINAL



```
# print(type(b))

print(2.5)
print(type(2.5))

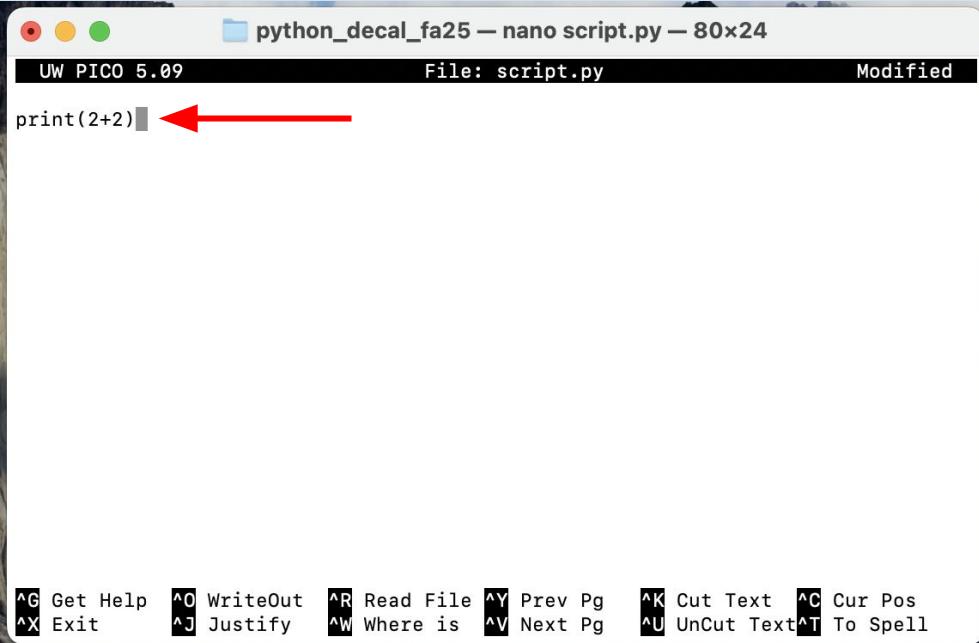
c = [0, 1, 2]
print(c)
[print(type(c))]

bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatype.py    script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
```

We will edit this new file and run it again on the terminal

Type: nano script.py





A screenshot of a terminal window titled "python_decal_fa25 – nano script.py – 80x24". The window shows the command "print(2+2)" on a single line. A red arrow points to the beginning of the line. The status bar at the bottom lists various nano editor keyboard shortcuts.

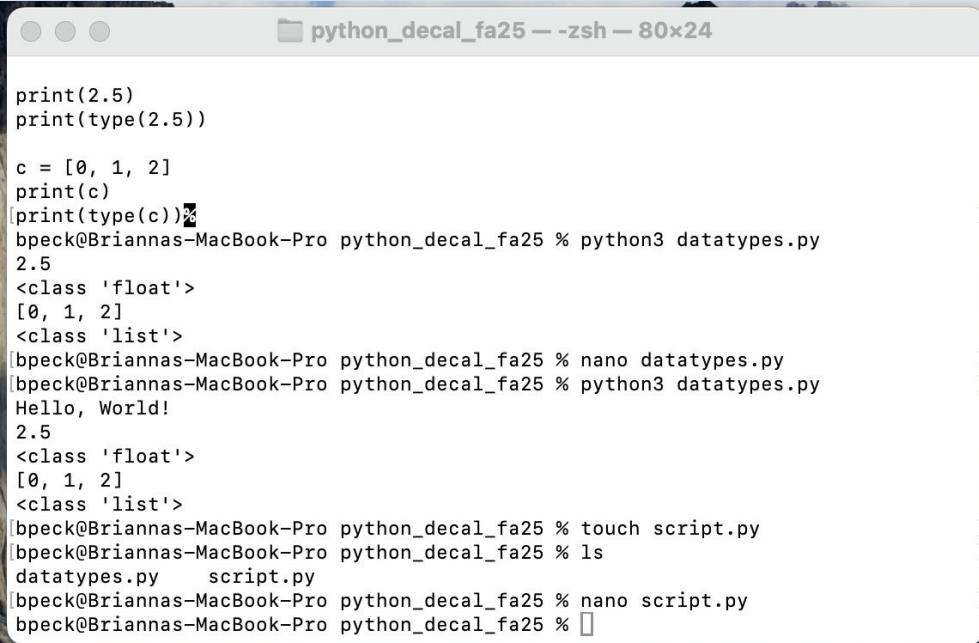
```
python_decal_fa25 – nano script.py – 80x24
UW PICO 5.09 File: script.py Modified
print(2+2) ←
```

^G Get Help ^O WriteOut ^R Read File ^Y Prev Pg ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where is ^V Next Pg ^U UnCut Text ^T To Spell

Add the following line to your script: print(2+2)

Then:

- Save your changes
- Close nano



A screenshot of a terminal window titled "python_decal_fa25 --zsh-- 80x24". The window displays the following Python script execution:

```
print(2.5)
print(type(2.5))

c = [0, 1, 2]
print(c)
print(type(c))
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatypes.py    script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

This is what your screen will output too





MAKE A NEW FILE ON TERMINAL

```
python_decal_fa25 --zsh-- 80x24

print(2.5)
print(type(2.5))

c = [0, 1, 2]
print(c)
print(type(c))
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatypes.py    script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
```

We will call Python once again on this file too

Type: [python3 script.py](#)





MAKE A NEW FILE ON TERMINAL

Python just did math for us!

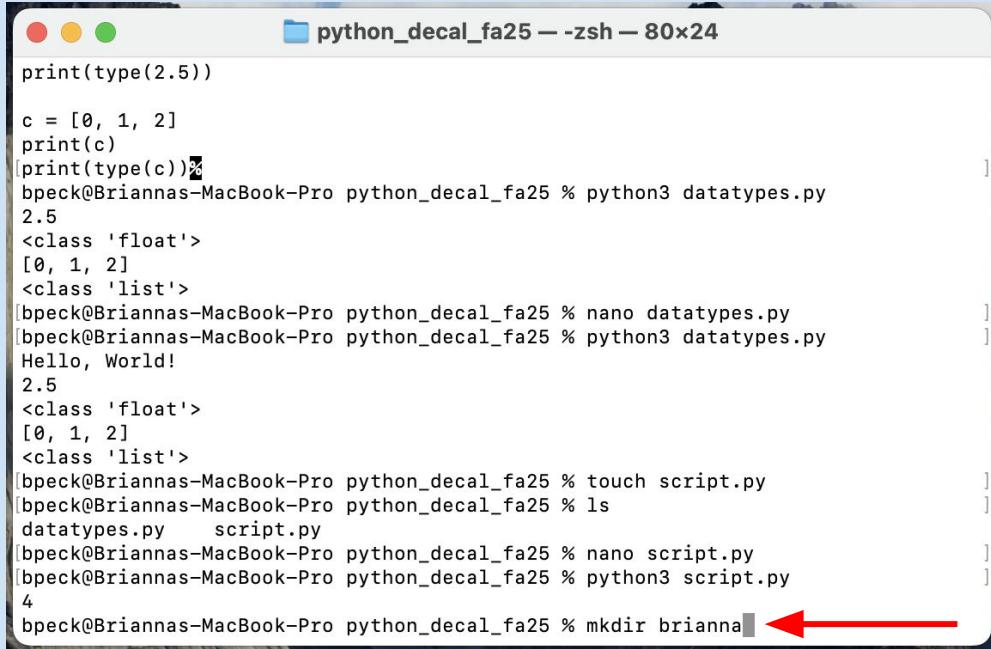
```
python_decal_fa25 --zsh -- 80x24

print(type(2.5))

c = [0, 1, 2]
print(c)
print(type(c))
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatypes.py    script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4 ←
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```



MAKE A NEW FOLDER



```
print(type(2.5))

c = [0, 1, 2]
print(c)
print(type(c))
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatypes.py    script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
bpeck@Briannas-MacBook-Pro python_decal_fa25 % mkdir brianna
```

To make a new folder on the terminal, we will use the command: mkdir

Type: mkdir <yourname>

Replace <yourname> with...
well... your name. So I would write:

mkdir brianna



MAKE A NEW FOLDER



```
c = [0, 1, 2]
print(c)
print(type(c))
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatypes.py    script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % mkdir brianna
[bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

This will be another super important folder while you are taking this class

mkdir stands for “make directory”





LIST CONTENTS



```
c = [0, 1, 2]
print(c)
print(type(c))
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatypes.py    script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % mkdir brianna
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls]
```

Now when we call ls on in our current working directory...

How many items should we now see?

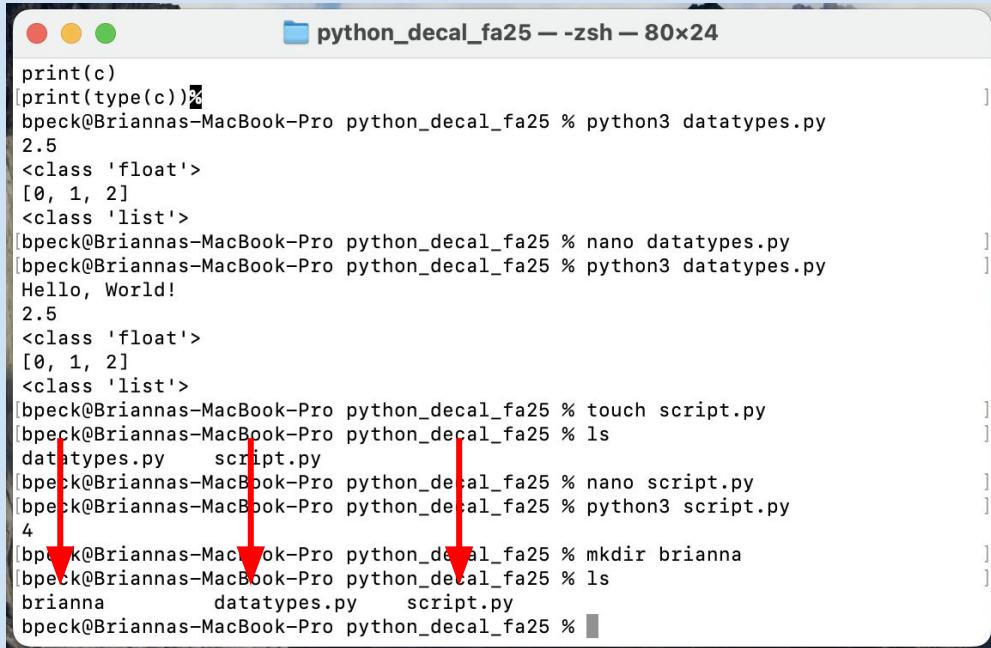
How many files?

How many folders?



LIST CONTENTS

We see three items!



A screenshot of a macOS terminal window titled "python_decal_fa25 --zsh-- 80x24". The window shows the following command-line session:

```
print(c)
print(type(c))
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatypes.py    script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
bpeck@Briannas-MacBook-Pro python_decal_fa25 % mkdir brianna
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
brianna      datatypes.py    script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 %
```

Three red arrows point from the bottom of the slide towards the "script.py" and "datatypes.py" files in the terminal output, highlighting them.

Two files:

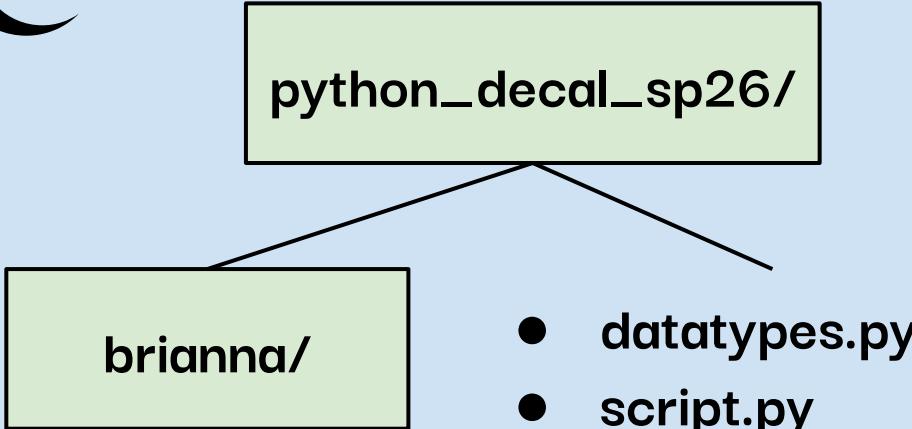
- **datatypes.py**
- **script.py**

One folder:

- <yourname>



QUICK NOTE: DIRECTORY TREES



Because we added a new folder,
our directory tree now looks like:

We have a folder and two files in
our python_decal_sp26
directory

CHANGE DIRECTORIES

```
print(c)
print(type(c))
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatypes.py    script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
bpeck@Briannas-MacBook-Pro python_decal_fa25 % mkdir brianna
bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
brianna      datatypes.py    script.py
bpeck@Briannas-MacBook-Pro python_decal_fa25 % cd brianna
```

To move into our new directory, we will call: cd

Type: cd <yourname>



CHANGE DIRECTORIES



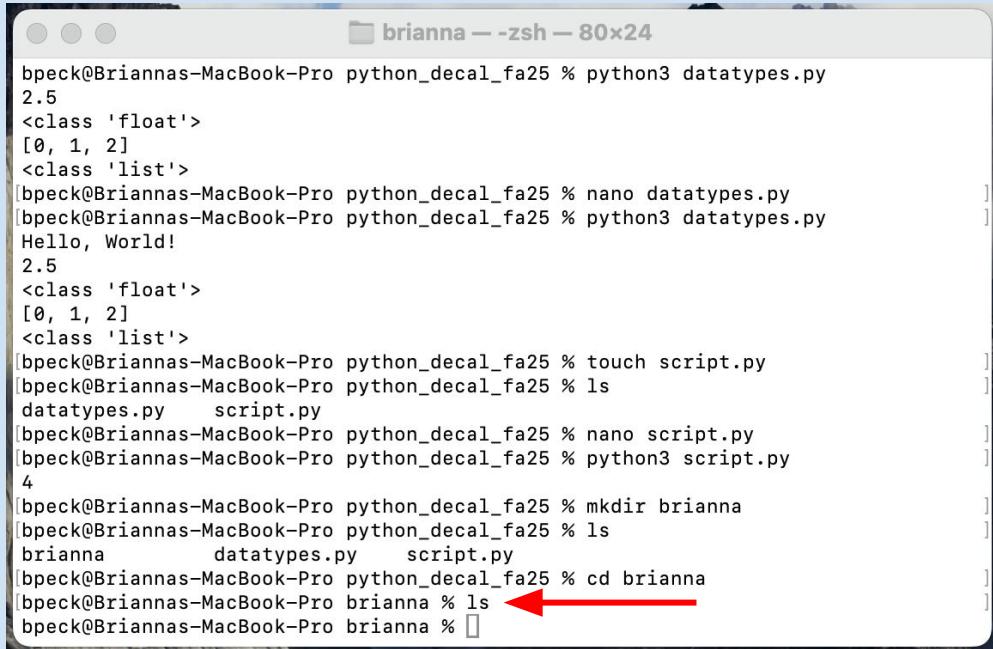
```
brianna — -zsh — 80x24
[1] $ print(type(c))
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[1] $ bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[2] $ bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[3] $ bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
[4] $ bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatypes.py      script.py
[5] $ bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
[6] $ bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
[7] $ bpeck@Briannas-MacBook-Pro python_decal_fa25 % mkdir brianna
[8] $ bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
brianna          datatypes.py      script.py
[9] $ bpeck@Briannas-MacBook-Pro python_decal_fa25 % cd brianna
[10] $ bpeck@Briannas-MacBook-Pro brianna %
```

Once again the folder next to the percentage sign changed, this might not be displayed on your terminal

If I call ls in this folder, will my terminal display anything?



LIST CONTENTS

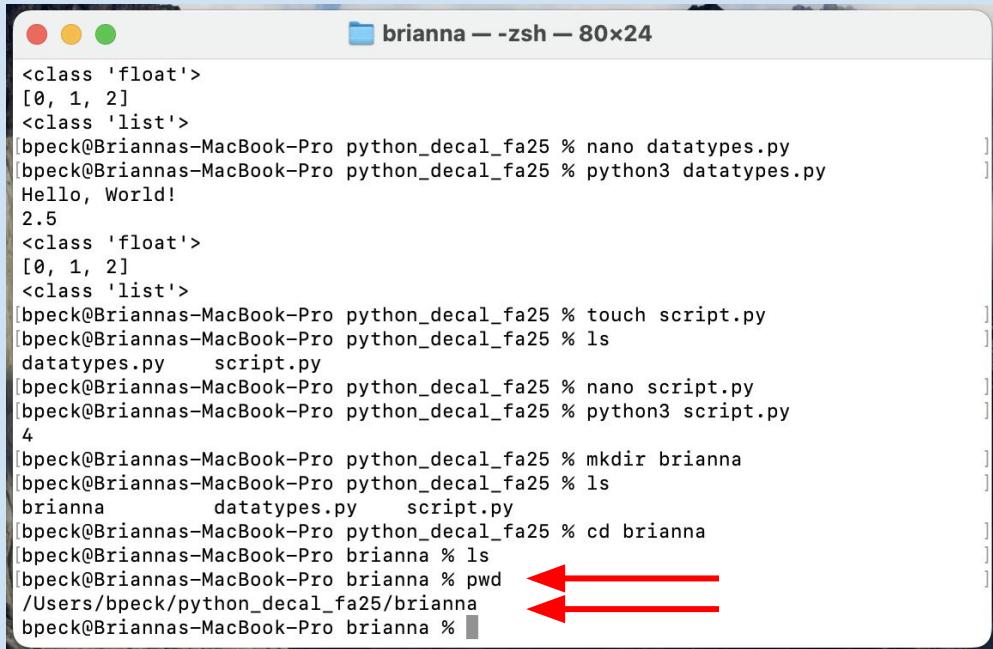


```
bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatypes.py    script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % mkdir brianna
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
brianna      datatypes.py    script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % cd brianna
[bpeck@Briannas-MacBook-Pro brianna % ls ←
[bpeck@Briannas-MacBook-Pro brianna % ]
```

Nope! It's a brand new folder so it is completely empty

What will our file path look like now? Type: pwd

PRINT WORKING DIRECTORY



```
brianna -- -zsh -- 80x24

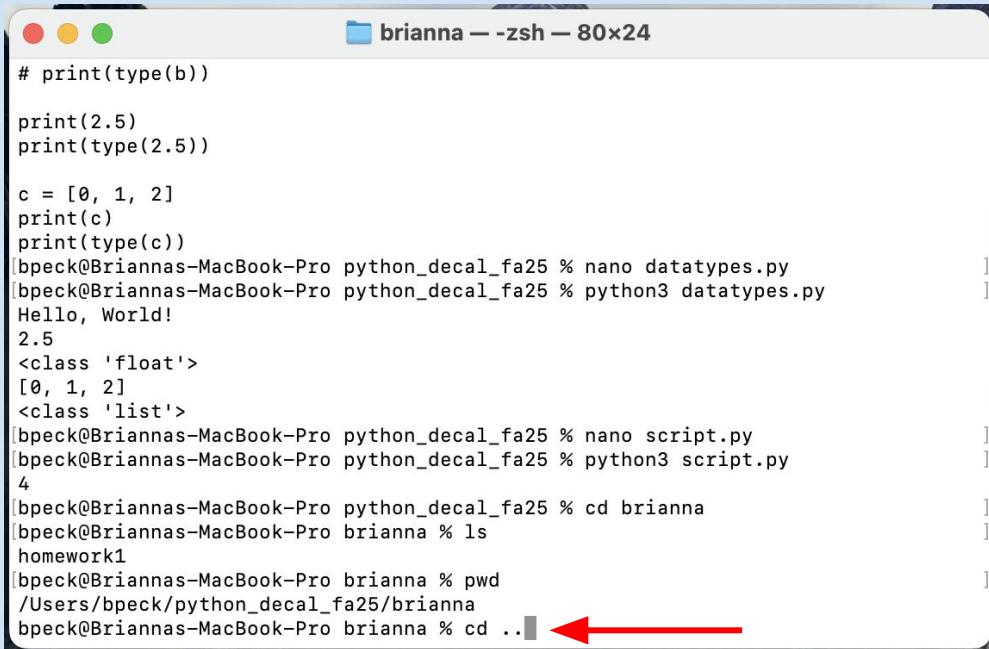
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % touch script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
datatypes.py    script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % mkdir brianna
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % ls
brianna      datatypes.py    script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % cd brianna
[bpeck@Briannas-MacBook-Pro brianna % ls
[bpeck@Briannas-MacBook-Pro brianna % pwd
/Users/bpeck/python_decal_fa25/brianna
[bpeck@Briannas-MacBook-Pro brianna %
```

Now we have brianna, inside of python_decal_sp26, inside of bpeck, inside of /Users/

You will use this folder as part of your homework

Parent Directory

To move up to a parent directory, use the command cd ..



```
# print(type(b))
print(2.5)
print(type(2.5))

c = [0, 1, 2]
print(c)
print(type(c))
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % cd brianna
[bpeck@Briannas-MacBook-Pro brianna % ls
homework1
[bpeck@Briannas-MacBook-Pro brianna % pwd
/Users/bpeck/python_decal_fa25/brianna
[bpeck@Briannas-MacBook-Pro brianna % cd ..]
```

A red arrow points to the command `cd ..` at the bottom of the terminal window.

Parent directory

Now if we call pwd

The screenshot shows a terminal window with the following content:

```
python_decal_fa25 --zsh-- 80x24

print(2.5)
print(type(2.5))

c = [0, 1, 2]
print(c)
print(type(c))
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % cd brianna
[bpeck@Briannas-MacBook-Pro brianna % ls
homework1
[bpeck@Briannas-MacBook-Pro brianna % pwd
/Users/bpeck/python_decal_fa25/brianna
[bpeck@Briannas-MacBook-Pro brianna % cd ..
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % pwd
```

A red arrow points to the last line of the terminal output, which is "bpeck@Briannas-MacBook-Pro python_decal_fa25 % pwd".

Parent Directory

```
python_decal_fa25 --zsh -- 80x24
print(type(2.5))

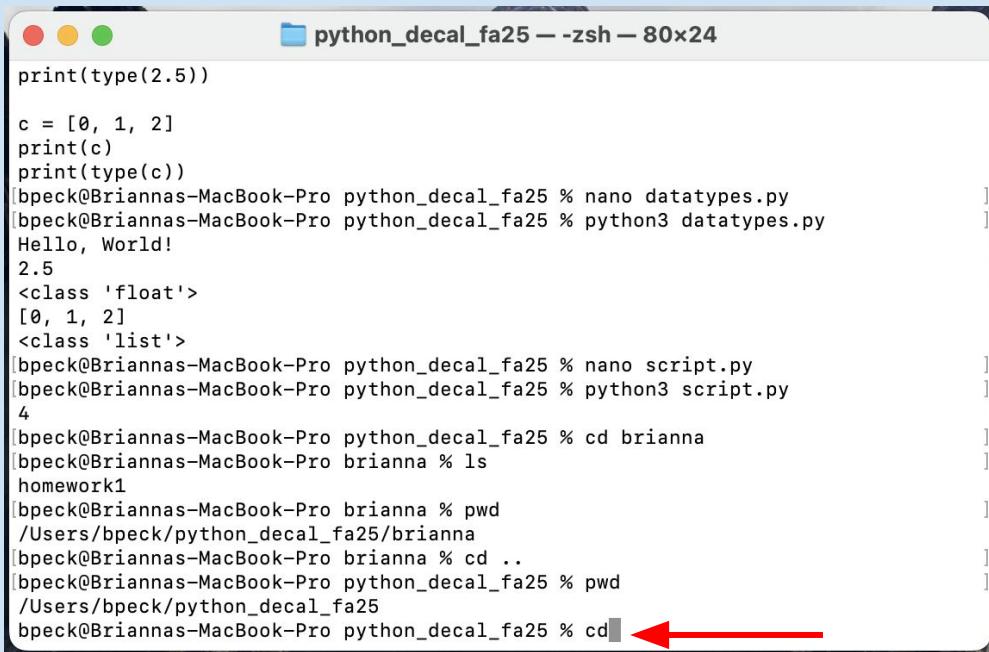
c = [0, 1, 2]
print(c)
print(type(c))
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % cd brianna
[bpeck@Briannas-MacBook-Pro brianna % ls
homework1
[bpeck@Briannas-MacBook-Pro brianna % pwd
/Users/bpeck/python_decal_fa25/brianna
[bpeck@Briannas-MacBook-Pro brianna % cd ..
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % pwd
/Users/bpeck/python_decal_fa25 ←
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % ]
```

Our file path changed and we are no longer inside of the yourname folder

Instead we are in
python_decal_sp26 again

BACK TO HOME DIRECTORY

To go all the way back to your home directory, you can call the command cd



A screenshot of a macOS terminal window titled "python_decal_fa25 --zsh-- 80x24". The window contains the following text:

```
print(type(2.5))

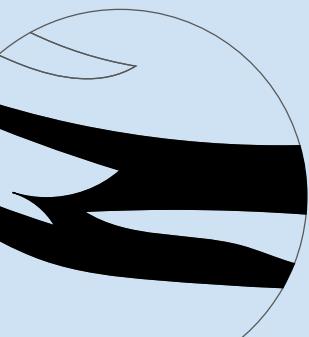
c = [0, 1, 2]
print(c)
print(type(c))
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % cd brianna
[bpeck@Briannas-MacBook-Pro brianna % ls
homework1
[bpeck@Briannas-MacBook-Pro brianna % pwd
/Users/bpeck/python_decal_fa25/brianna
[bpeck@Briannas-MacBook-Pro brianna % cd ..
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % pwd
/Users/bpeck/python_decal_fa25
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % cd
```

A red arrow points to the last "cd" command at the bottom of the terminal window.

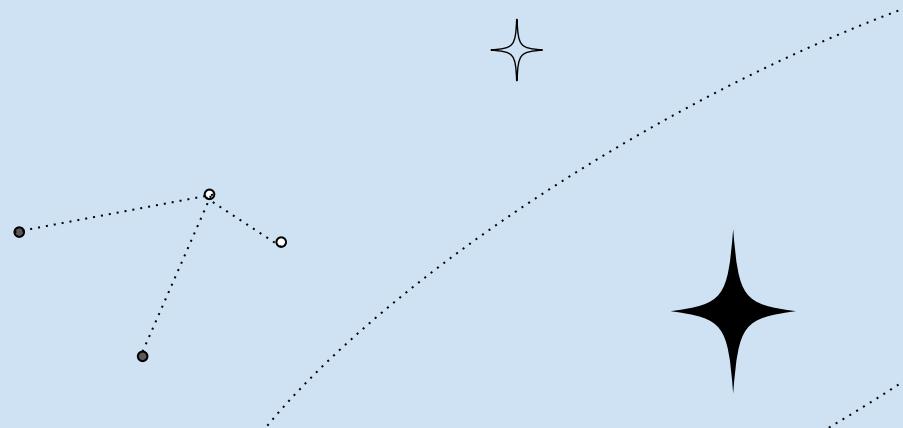
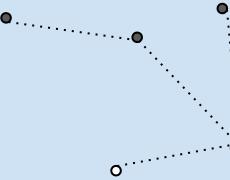
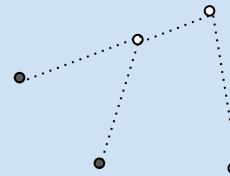
BACK TO HOME DIRECTORY

```
bpeck -- zsh -- 80x24
print(c)
print(type(c))
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano datatypes.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 datatypes.py
Hello, World!
2.5
<class 'float'>
[0, 1, 2]
<class 'list'>
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % nano script.py
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % python3 script.py
4
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % cd brianna
[bpeck@Briannas-MacBook-Pro brianna % ls
homework1
[bpeck@Briannas-MacBook-Pro brianna % pwd
/Users/bpeck/python_decal_fa25/brianna
[bpeck@Briannas-MacBook-Pro brianna % cd ..
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % pwd
/Users/bpeck/python_decal_fa25
[bpeck@Briannas-MacBook-Pro python_decal_fa25 % cd
[bpeck@Briannas-MacBook-Pro ~ % pwd ←
/Users/bpeck ←
[bpeck@Briannas-MacBook-Pro ~ % ]
```

Now when we call pwd, we are back in the home directory like when we first opened the terminal



Let's Practice





PRACTICE QUESTION

Close those laptops!

Define each of these commands in your own words:

- **pwd**
- **ls**
- **cat**
- **cd**
- **nano**





PRACTICE QUESTION Answer

- **pwd** = prints your the folder you are working in
- **ls** = lists files and folders inside of you current folder
- **cat** = shows the inside of a file on the command line
- **cd** = allows you to move from folder to folder
- **nano** = allows you to open up a file and edit it





* 04 *

Lecture Engagement

Show us what you know!



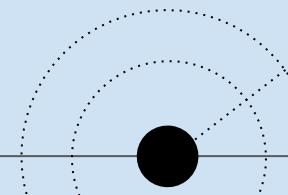
LECTURE ENGAGEMENT

Fill out the Lecture Check!

Linked [here](#).

Always due two days after class.

Takes 1-5 minutes to complete.



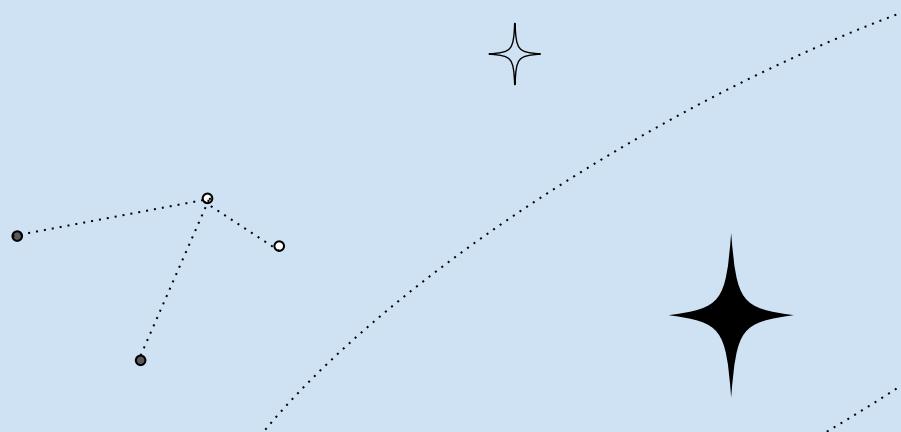
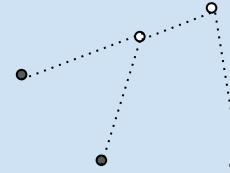
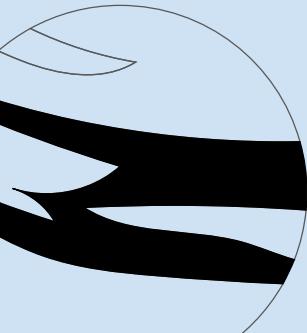


* 05 *

Bonus Content

For helping with homework

PROBLEM #1





MAKE ANOTHER A NEW FOLDER

A screenshot of a Mac OS X terminal window titled "bpeck -- zsh -- 80x24". The window shows the command "cd python_decal_fa25/brianna" being typed. A red arrow points to the right side of the command line, indicating where the user should type the command.

```
Last login: Fri Jul 25 09:15:02 on ttys006  
bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25/brianna
```

In Problem #1 of HW #1 you are asked to make a new folder called homework1 in the yourname directory.

Start by opening your terminal and navigating to the yourname folder.



brianna -- zsh -- 80x24

```
Last login: Fri Jul 25 11:26:47 on ttys006
bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25/brianna
bpeck@Briannas-MacBook-Pro brianna % mkdir homework1
```

To make a new folder inside
of the yourname folder, type:
mkdir homework1





LIST CONTENTS

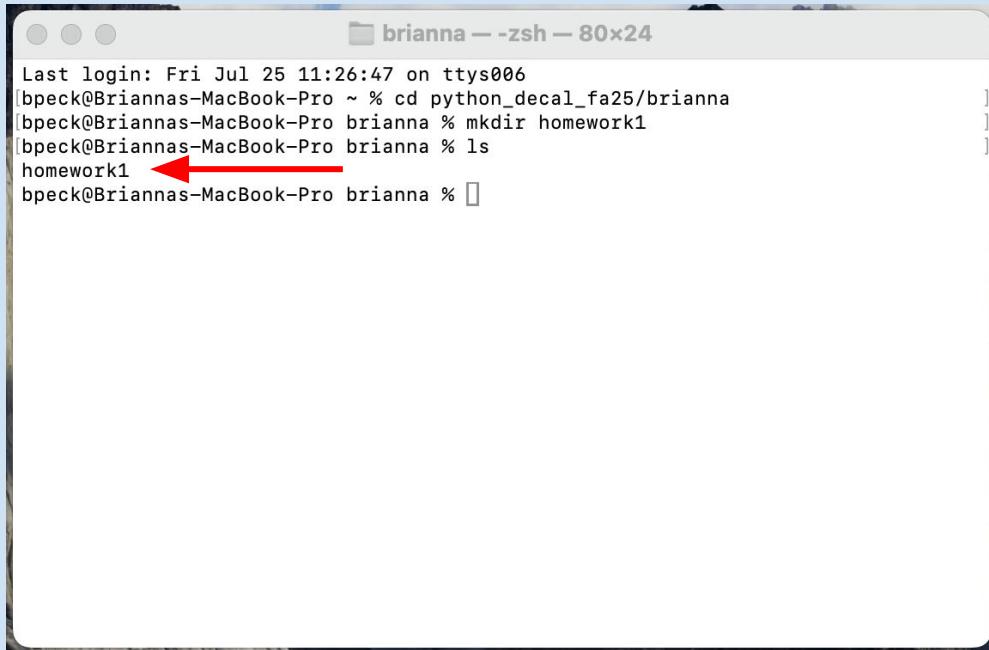


```
Last login: Fri Jul 25 11:26:47 on ttys006
bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25/brianna
bpeck@Briannas-MacBook-Pro brianna % mkdir homework1
bpeck@Briannas-MacBook-Pro brianna % ls
```

To see the contents of the yourname directory, type `ls`



LIST CONTENTS

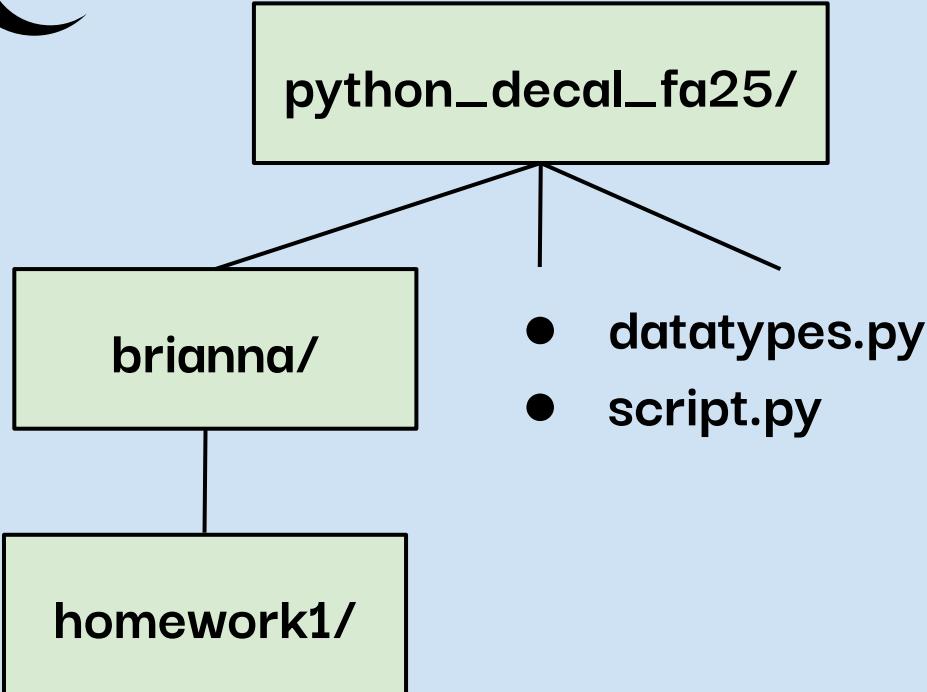


```
Last login: Fri Jul 25 11:26:47 on ttys006
bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25/brianna
bpeck@Briannas-MacBook-Pro brianna % mkdir homework1
bpeck@Briannas-MacBook-Pro brianna % ls
homework1 ←
bpeck@Briannas-MacBook-Pro brianna %
```

You will now see a new folder called homework1



QUICK NOTE: DIRECTORY TREES



If you recall the directory tree from lecture, this is now what your directory tree looks like

So homework1 is a folder that lives inside of brianna which lives inside of python_decal_fa25



CHANGE DIRECTORIES

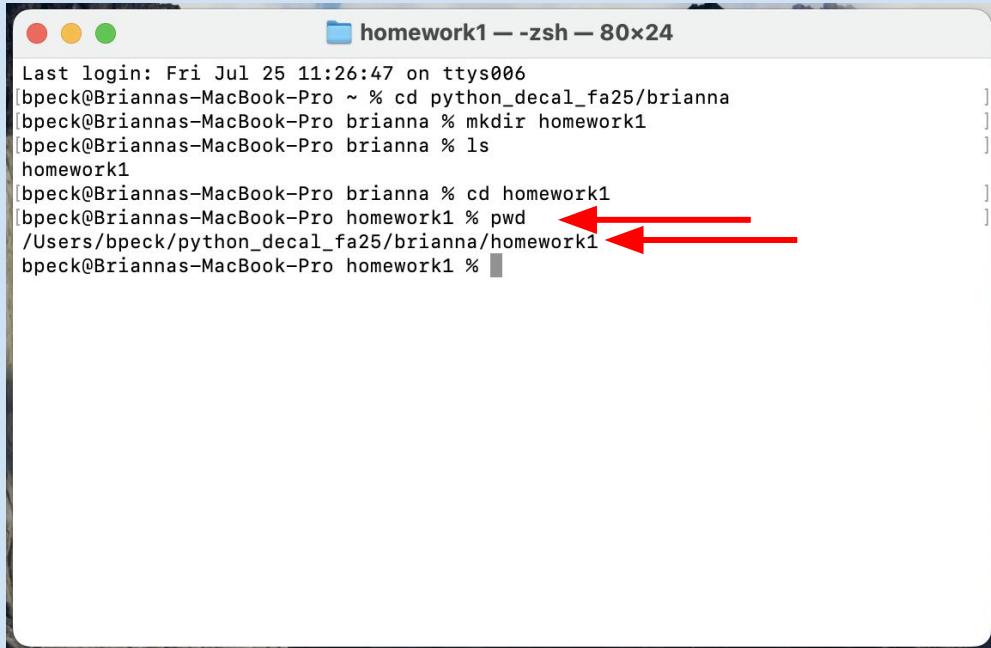


```
Last login: Fri Jul 25 11:26:47 on ttys006
bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25/brianna
bpeck@Briannas-MacBook-Pro brianna % mkdir homework1
bpeck@Briannas-MacBook-Pro brianna % ls
homework1
bpeck@Briannas-MacBook-Pro brianna % cd homework1
```

To move into homework1 we will use the command cd



PRINT WORKING DIRECTORY



A screenshot of a Mac OS X terminal window titled "homework1 — zsh — 80x24". The window shows a command-line session:

```
Last login: Fri Jul 25 11:26:47 on ttys006
bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25/brianna
bpeck@Briannas-MacBook-Pro brianna % mkdir homework1
bpeck@Briannas-MacBook-Pro brianna % ls
homework1
bpeck@Briannas-MacBook-Pro brianna % cd homework1
bpeck@Briannas-MacBook-Pro homework1 % pwd
/Users/bpeck/python_decal_fa25/brianna/homework1
bpeck@Briannas-MacBook-Pro homework1 %
```

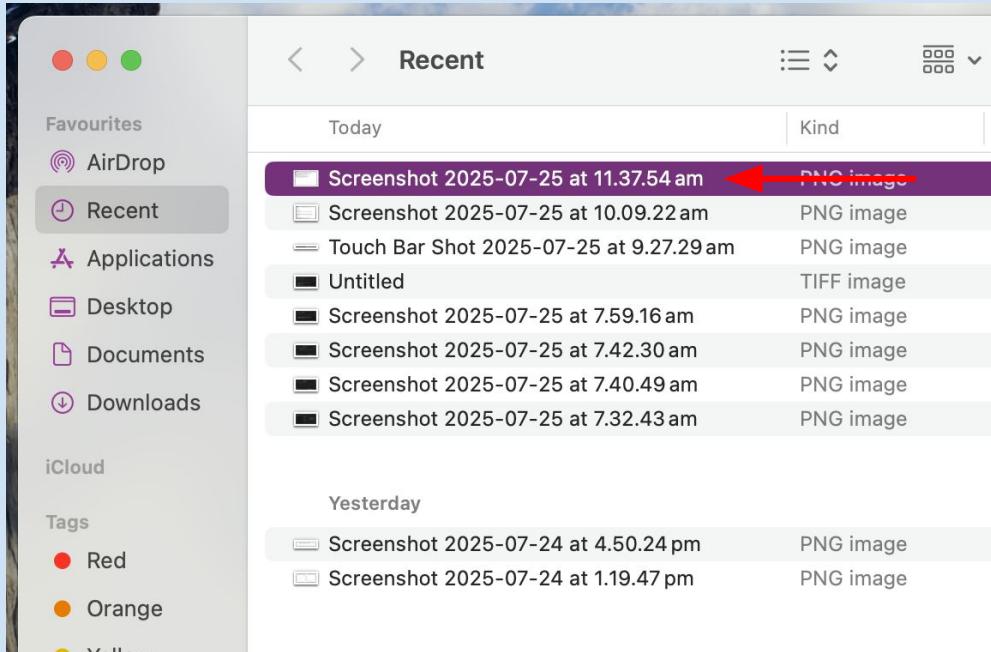
Two red arrows point to the command "pwd" and its output "/Users/bpeck/python_decal_fa25/brianna/homework1".

When we call pwd, which prints our current working directory

We see our entire file path

Next, you are asked to take a screenshot of this process. Your picture should look like the one to the left.

save screenshot



To save it in you homework1 folder, here is one way to do that after taking the screenshot.

Navigate to your GUI.

Find the screenshot in the Recents/ folder or the Screenshots/ folder



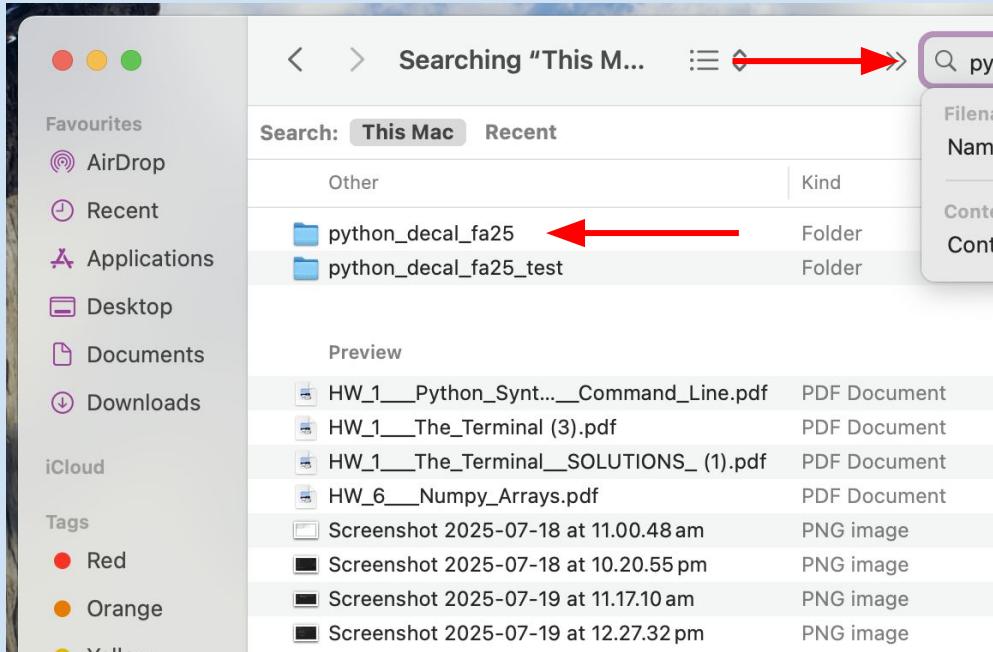
save screenshot



Drag the screenshot out
onto your desktop



save screenshot

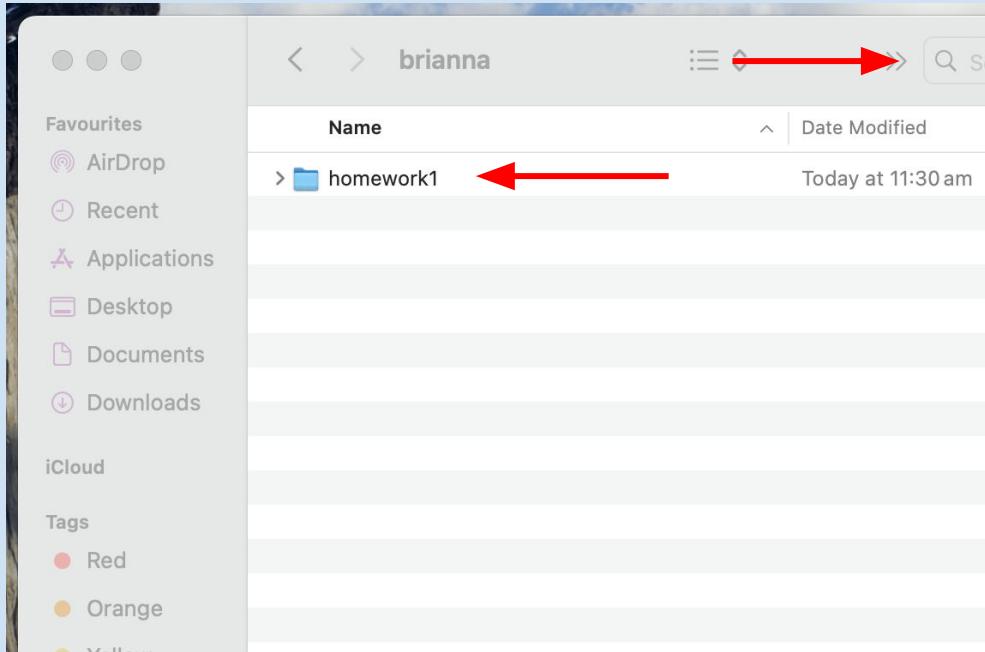


Search for the
python_decal_fa25 folder in
your GUI

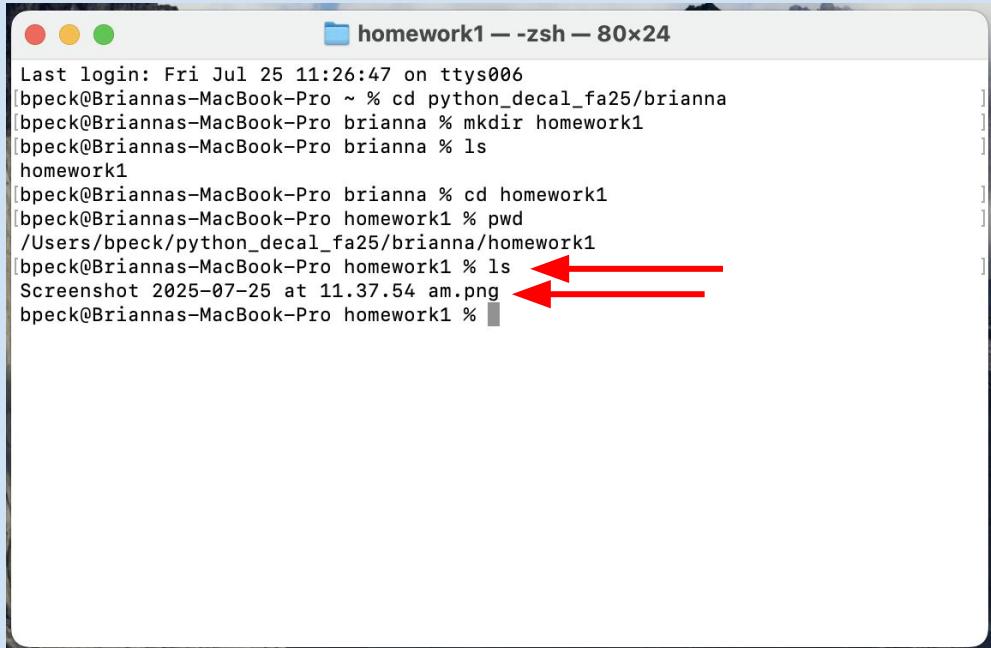
Once you have found it, open
up **yourname/homework1**

save screenshot

Drag and drop your
screenshot into this folder!



save screenshot



A screenshot of a macOS terminal window titled "homework1 — zsh — 80x24". The window shows the following command history:

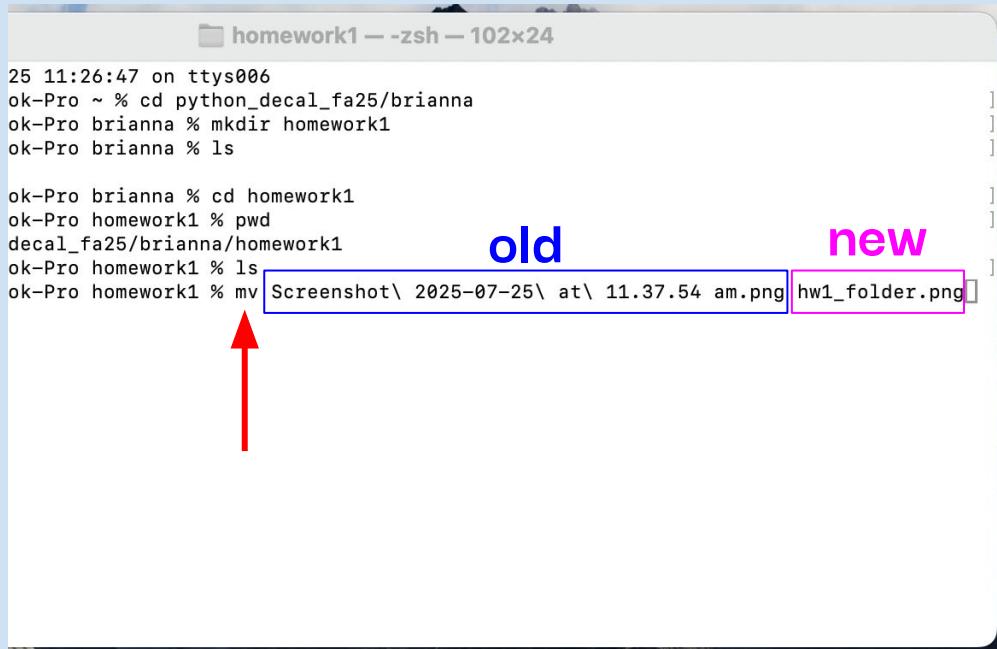
```
Last login: Fri Jul 25 11:26:47 on ttys006
[bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25/brianna
[bpeck@Briannas-MacBook-Pro brianna % mkdir homework1
[bpeck@Briannas-MacBook-Pro brianna % ls
homework1
[bpeck@Briannas-MacBook-Pro brianna % cd homework1
[bpeck@Briannas-MacBook-Pro homework1 % pwd
/Users/bpeck/python_decal_fa25/brianna/homework1
[bpeck@Briannas-MacBook-Pro homework1 % ls
Screenshot 2025-07-25 at 11.37.54 am.png
[bpeck@Briannas-MacBook-Pro homework1 %
```

Two red arrows point to the line "Screenshot 2025-07-25 at 11.37.54 am.png" in the terminal output.

You can confirm the screenshot is in your terminal by calling ls

We could have renamed the file earlier, but let's practice renaming it on the command line

Rename a FILE WITH MV



A terminal window titled "homework1 -- zsh -- 102x24" showing a command-line session. The session starts with the user navigating to their workspace and creating a new directory named "homework1". Then, they change into the "homework1" directory and list its contents. Finally, they use the "mv" command to rename a file from "Screenshot\ 2025-07-25\ at\ 11.37.54 am.png" to "hw1_folder.png". A red arrow points to the "mv" command, and the "old" name is highlighted with a blue box, while the "new" name is highlighted with a pink box.

```
25 11:26:47 on ttys006
ok-Pro ~ % cd python_decal_fa25/brianna
ok-Pro brianna % mkdir homework1
ok-Pro brianna % ls
ok-Pro brianna % cd homework1
ok-Pro homework1 % pwd
decal_fa25/brianna/homework1
ok-Pro homework1 % ls
ok-Pro homework1 % mv Screenshot\ 2025-07-25\ at\ 11.37.54 am.png hw1_folder.png
```

Use the mv command to rename your file (you can also use this to move files in general)

The syntax is:

mv <old_name> <new_name>



LIST CONTENTS

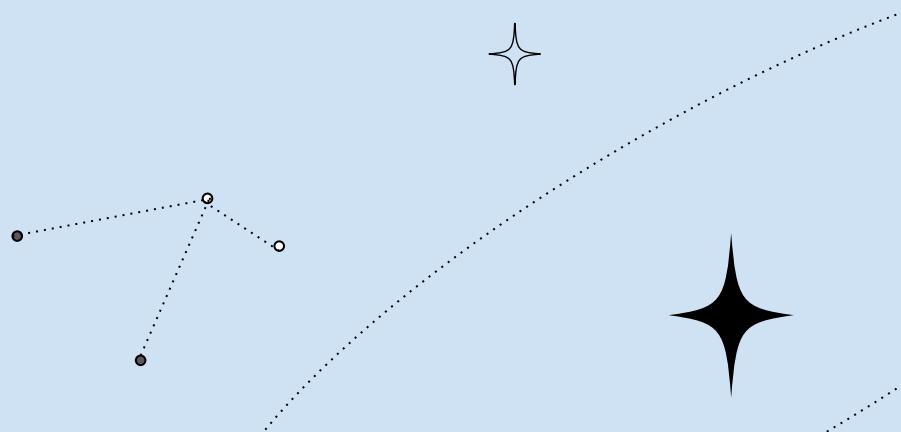
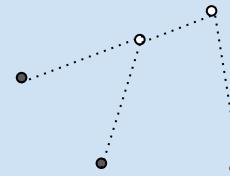
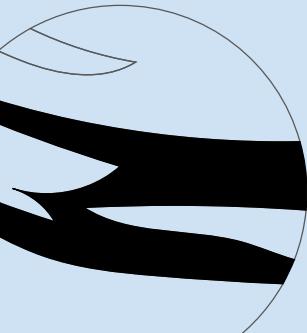


```
Last login: Fri Jul 25 11:26:47 on ttys006
[bpeck@Briannas-MacBook-Pro ~ % cd python_decal_fa25/brianna
[bpeck@Briannas-MacBook-Pro brianna % mkdir homework1
[bpeck@Briannas-MacBook-Pro brianna % ls
homework1
[bpeck@Briannas-MacBook-Pro brianna % cd homework1
[bpeck@Briannas-MacBook-Pro homework1 % pwd
/Users/bpeck/python_decal_fa25/brianna/homework1
[bpeck@Briannas-MacBook-Pro homework1 % ls
[bpeck@Briannas-MacBook-Pro homework1 % mv Screenshot\ 2025-07-25\ at\ 11
[bpeck@Briannas-MacBook-Pro homework1 % ls
hw1_folder.png ←
[bpeck@Briannas-MacBook-Pro homework1 % ]
```

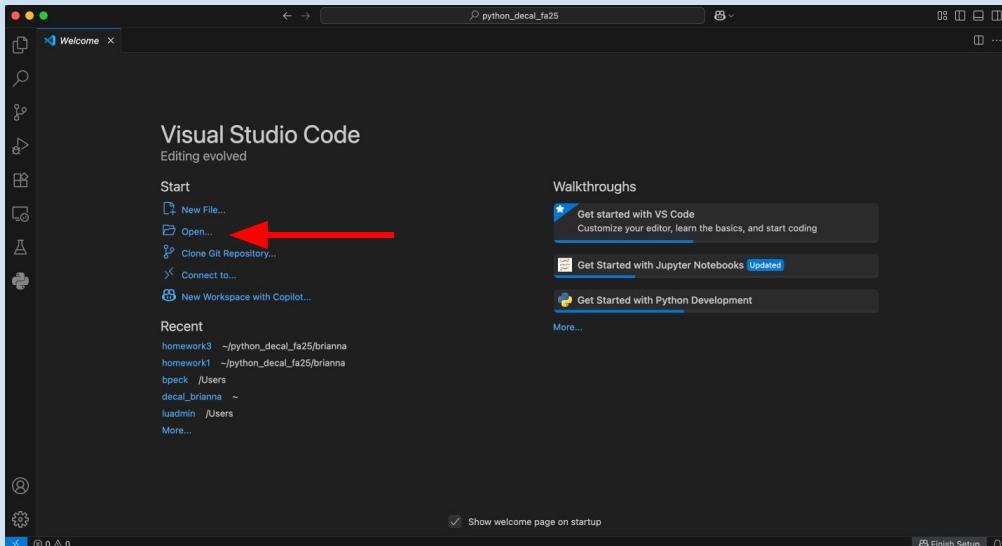
You can confirm the new renaming by calling `ls` on the command line



PROBLEM #2



OPEN VS CODE

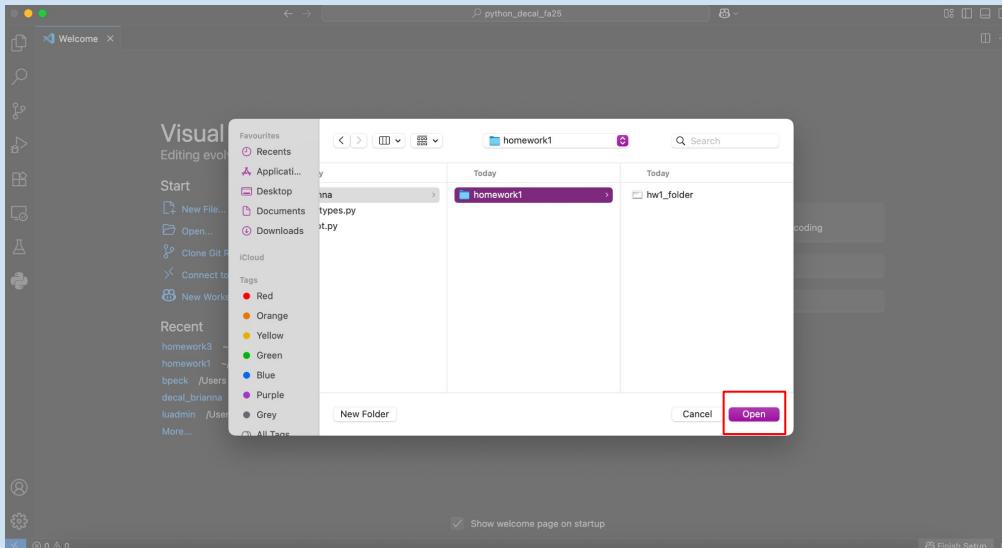


In Problem #2 of HW #1 you are asked to make a new file in VS Code.

To open a new file in VS Code, start by opening the application

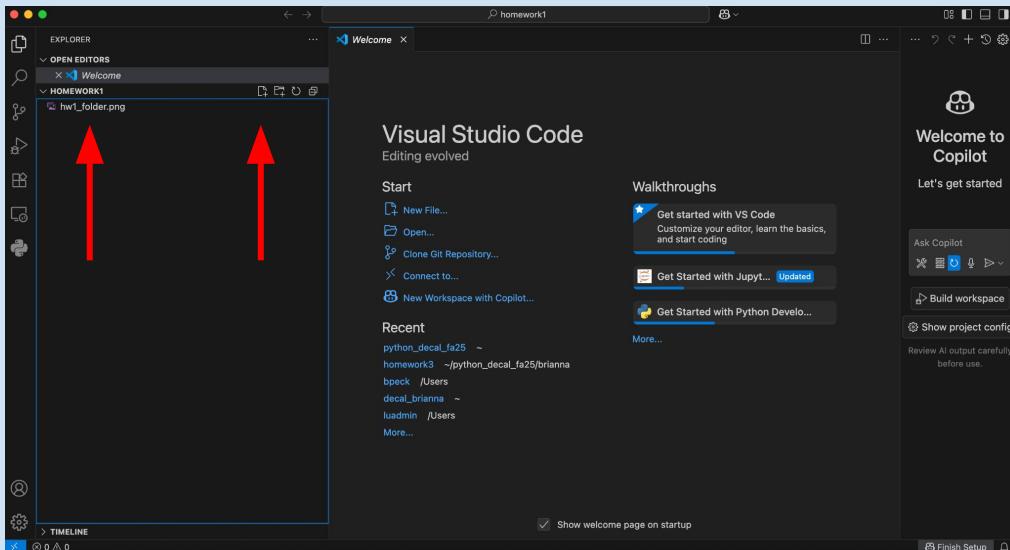
Then clicking on Open...

OPEN A NEW FILE



Navigate to the homework1/ folder you just made on the terminal

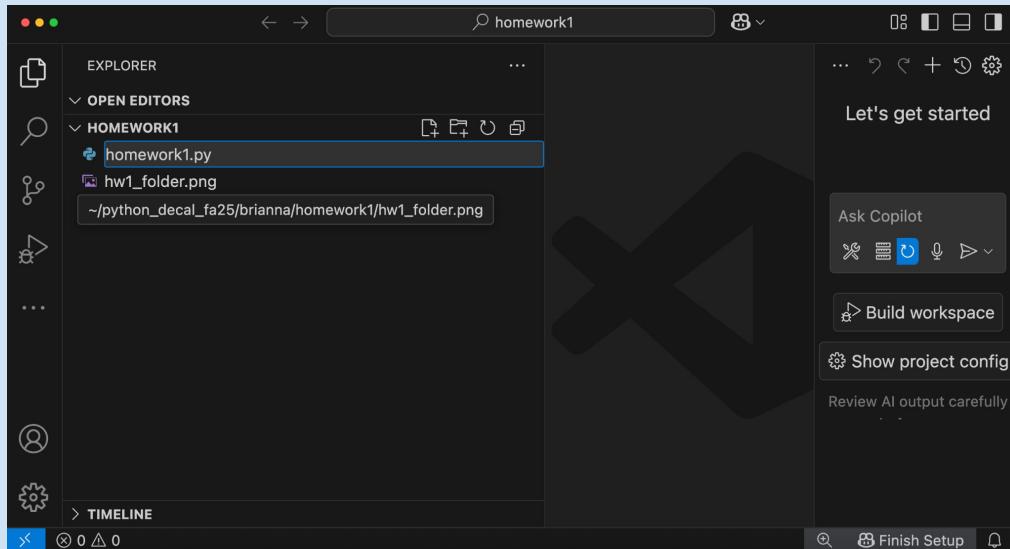
OPEN A NEW FILE



You now see the screenshot
we just renamed: hw1_folder

Then click on the New File...
icon the second red arrow is
pointing to

OPEN A NEW FILE

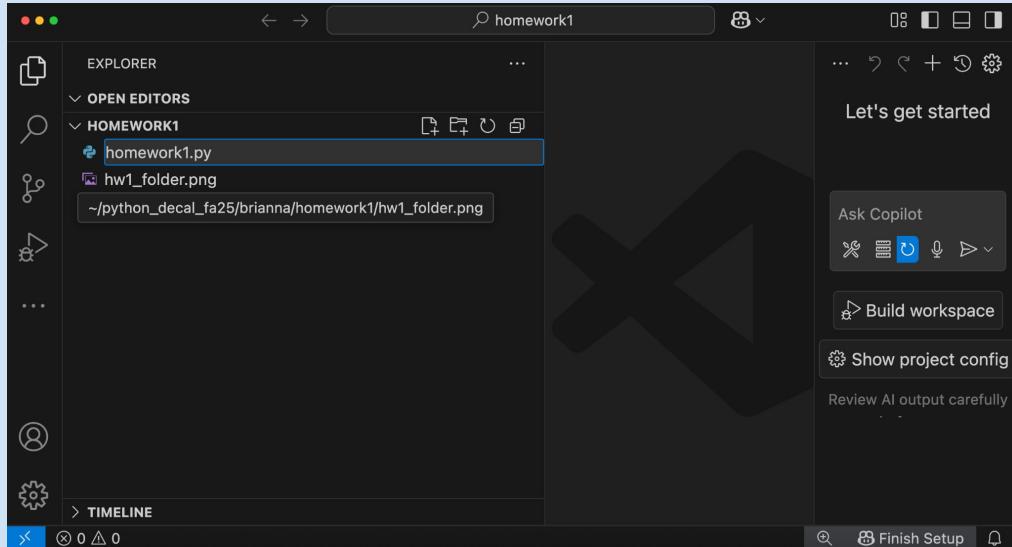


Name the new file:
homework1.py

Then start the other
problems!



MAKE ANOTHER A NEW FOLDER



Name the new file:
homework1.py

