Lab 1 Manual

Lab1 Activity

CSE8A Fall 2021 - Lab 1

Workspaces and Python on EdStem

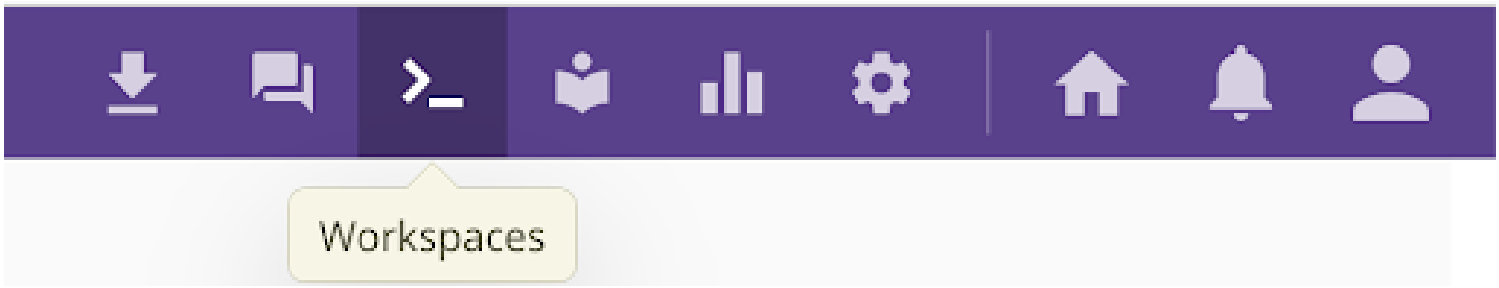
Learning Goal: Learn how Python works on EdStem

Part 1. Experimenting with Python [35 mins]

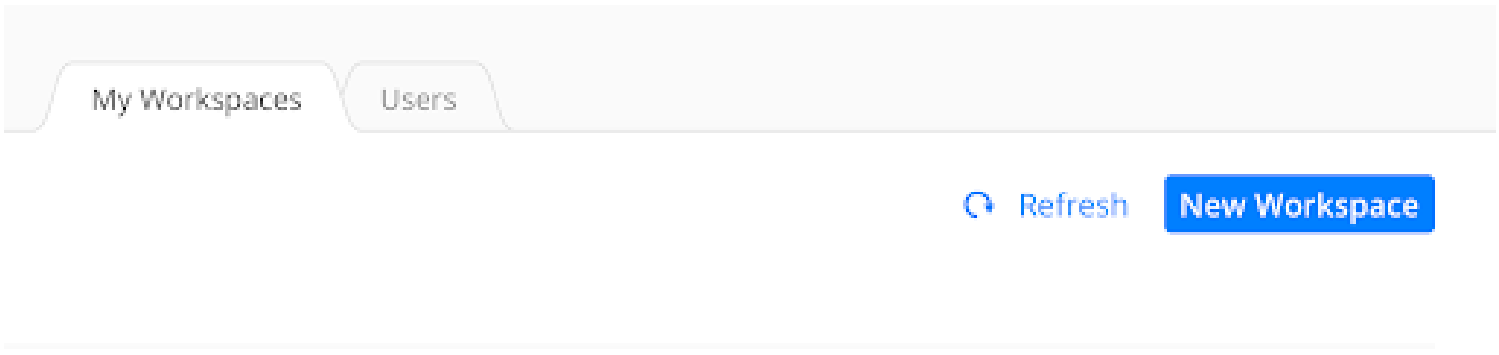
In this lab, you will experiment with Python and learn more about navigating the Ed platform. Feel free to ask tutors in your lab sessions any questions during the lab.

Creating a Workspace

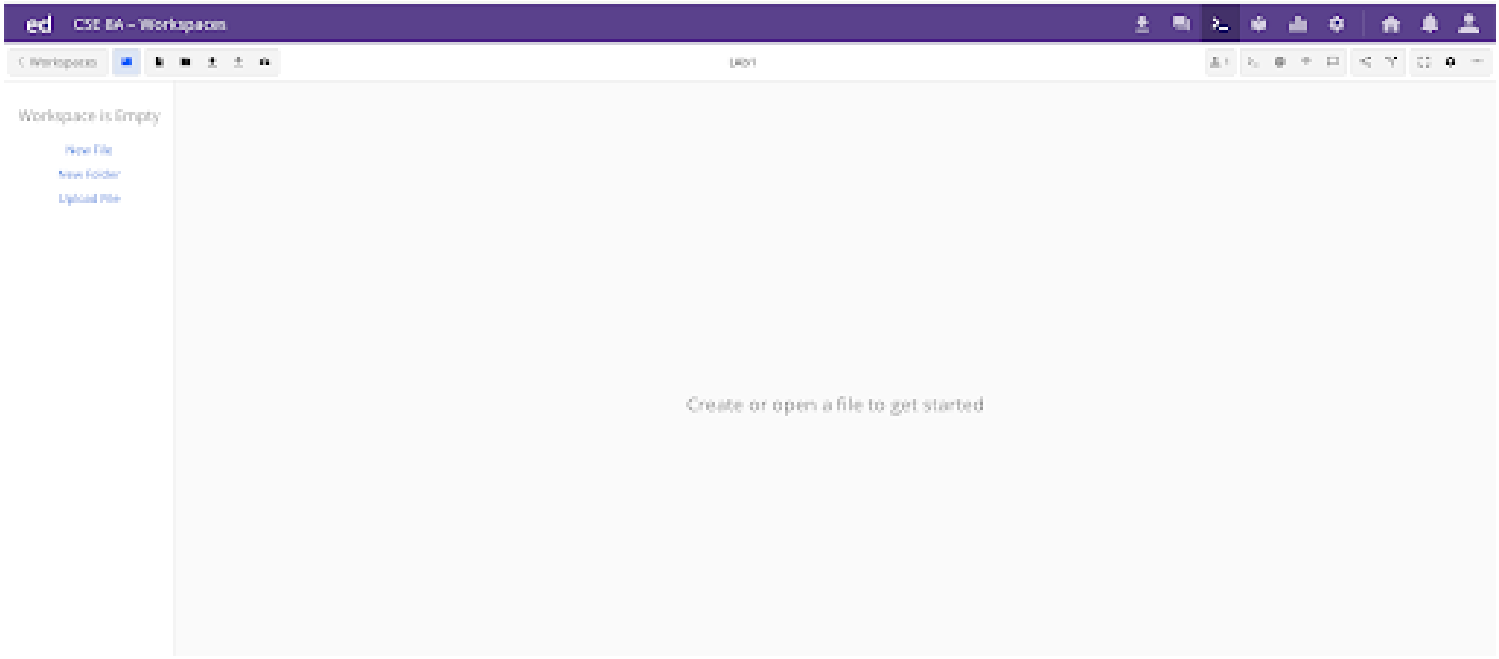
Open up our course in Ed (https://edstem.org/us/courses/14060) and click the Workspaces icon at the top. **(You may want to open up another EdStem window, so that you can refer to this lab manual while working on the following steps.)**



Then make a new Workspace with the “New Workspace” button – call it "Lab1"



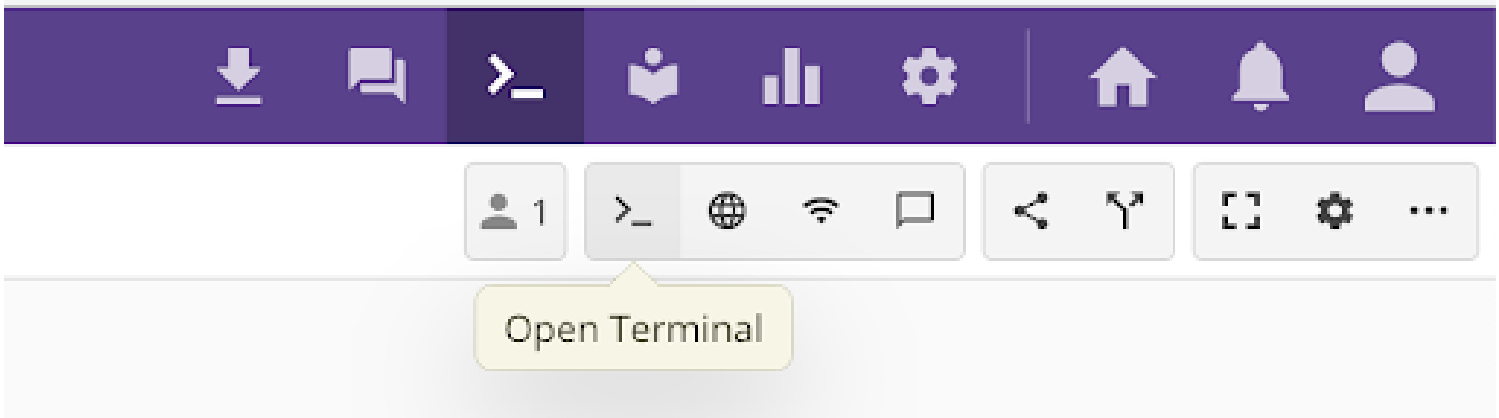
You should see a screen like this:



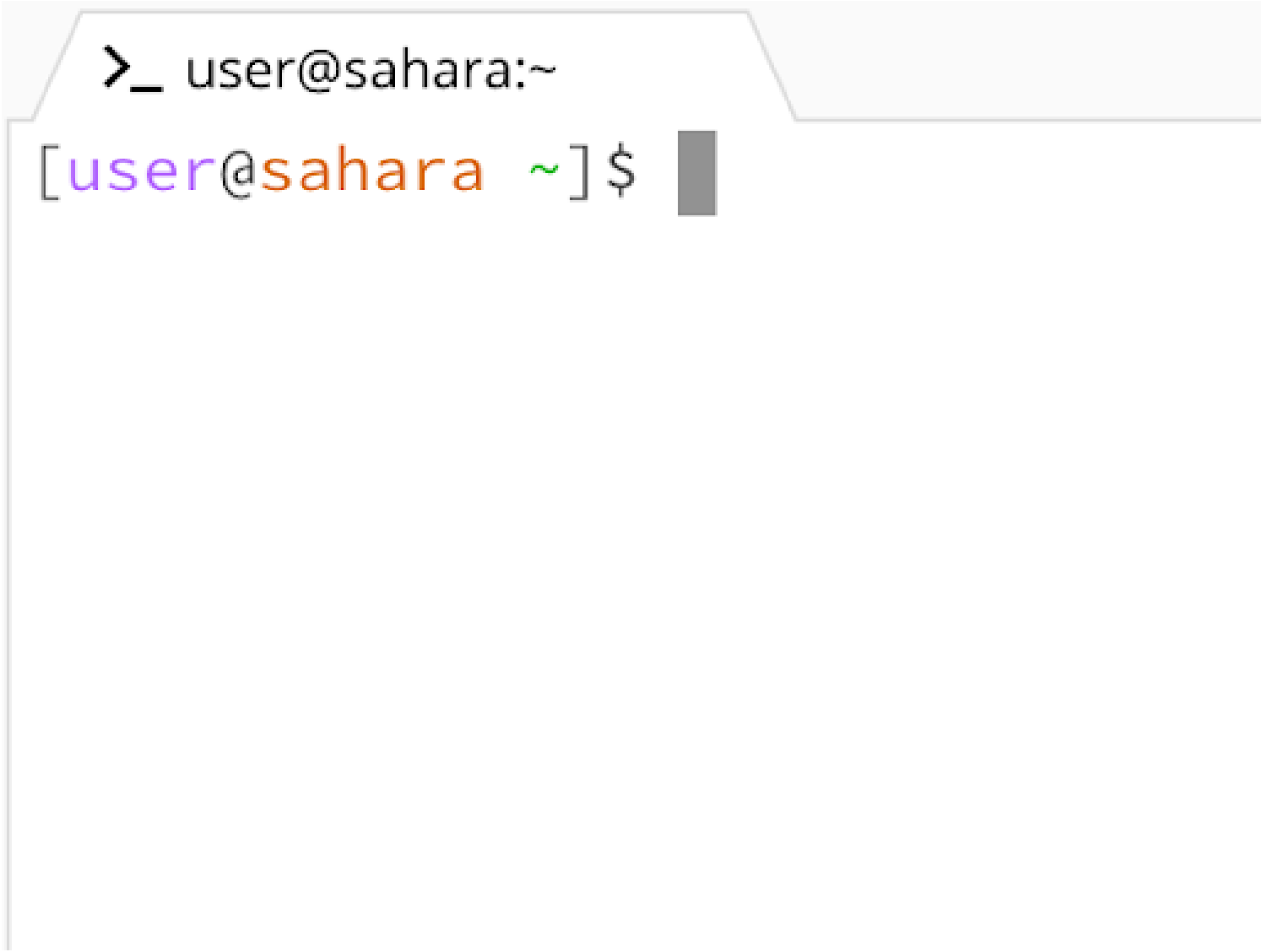
If you don't see this, speak up! We can help!

Starting Interactive Python

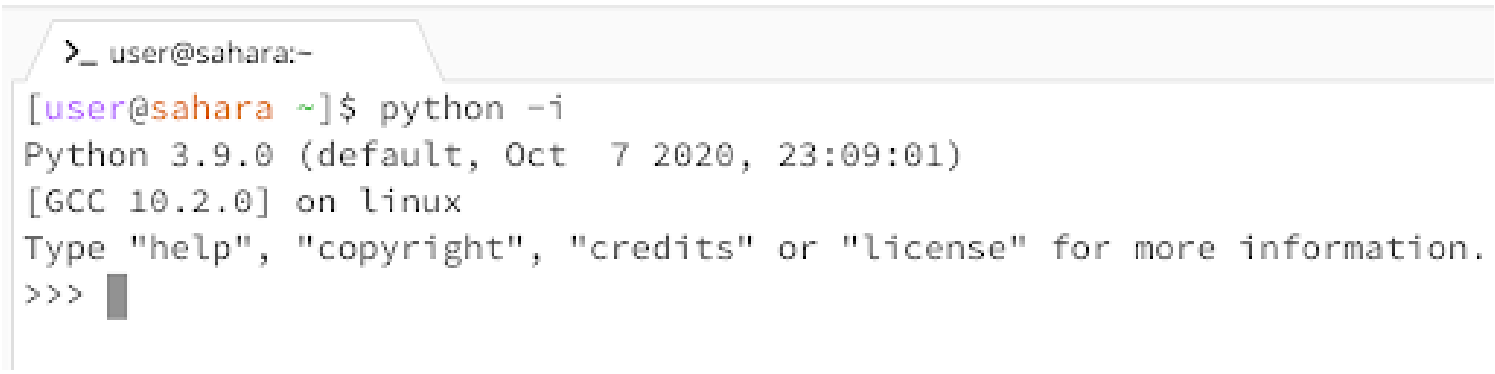
Click the “Open Terminal” button (it looks like the Workspaces button but smaller and gray):



You should see something like this:



Next, type in `python -i` and press Enter.



You've started Python, and can now start typing in Python expressions (what does that mean? Read on!)

Experimenting with Python

This screenshot is from the first section of the Stepik textbook (https://stepik.org/lesson/511146/step/1?unit=503344)

At a Python prompt (after the `>>>` in IDLE or at a terminal after running `python`), try typing a number and pressing enter. Here's what you should expect to see:

```
>>> 4
4
```

We typed in a number, and Python printed it back out at us. Let's try with arithmetic:

```
>>> 4 + 3
7

>>> 4 + 3 + 9
16
```

This is showing one of the most familiar tasks we can accomplish with Python – we can use it as a calculator! Python is equipped with many familiar arithmetic operations, like *, /, +, and -. It also evaluates operations in parentheses first, like we may expect from math:

```
>>> 4 * 8 - 2
30
>>> 4 * (8 - 2)
24
```

We can *nest* the parentheses and expressions as much as we like, and Python will calculate the answer for us:

```
>>> 4 * (8 - 2) + 3  * (99 / (2 + 1))
123.0
```

Try typing in all of those examples yourself! Then, try the following experiments:

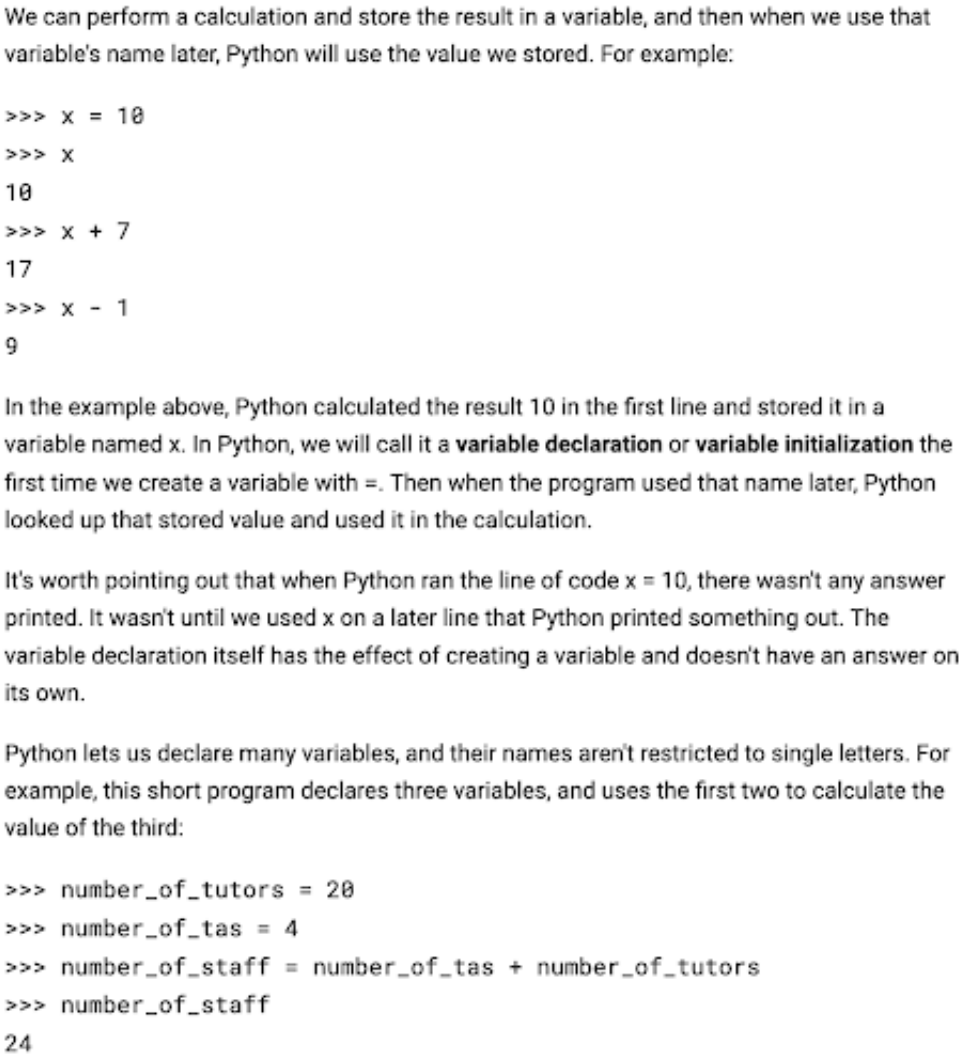
- Try dividing an odd number by an even number
- Try to cause an error. Deliberately make a mistake in one of the examples, like missing a parenthesis or forgetting an operator.

Think about these questions as you perform different experiments. Feel free to ask tutors any questions!

- What is a lesson you learned about Python as a result of your experiments?
- What is a question you have about Python as a result of your experiments?
- Did you think of ways to make an error with arithmetic?
- What error messages did you produce?

Defining Variables

Here's a screenshot from the next subsection of the textbook (https://stepik.org/lesson/511146/step/3?unit=503344).



We can perform a calculation and store the result in a variable, and then when we use that variable's name later, Python will use the value we stored. For example:

```
>>> x = 10
>>> x
10
>>> x + 7
17
>>> x - 1
9
```

In the example above, Python calculated the result 10 in the first line and stored it in a variable named x. In Python, we will call it a **variable declaration** or **variable initialization** the first time we create a variable with =. Then when the program used that name later, Python looked up that stored value and used it in the calculation.

It's worth pointing out that when Python ran the line of code x = 10, there wasn't any answer printed. It wasn't until we used x on a later line that Python printed something out. The variable declaration itself has the effect of creating a variable and doesn't have an answer on its own.

Python lets us declare many variables, and their names aren't restricted to single letters. For example, this short program declares three variables, and uses the first two to calculate the value of the third:

```
>>> number_of_tutors = 20
>>> number_of_tas = 4
>>> number_of_staff = number_of_tas + number_of_tutors
>>> number_of_staff
24
```

Type in all of these examples in Python to try them out, like what you did for the numeric examples.

Then, define at least two of your own variables with different names and initialize them to different values.

Variables and Quitting Python

You can quit interactive Python by typing `exit()` and pressing Enter (you can also press Control-D for a keyboard shortcut):

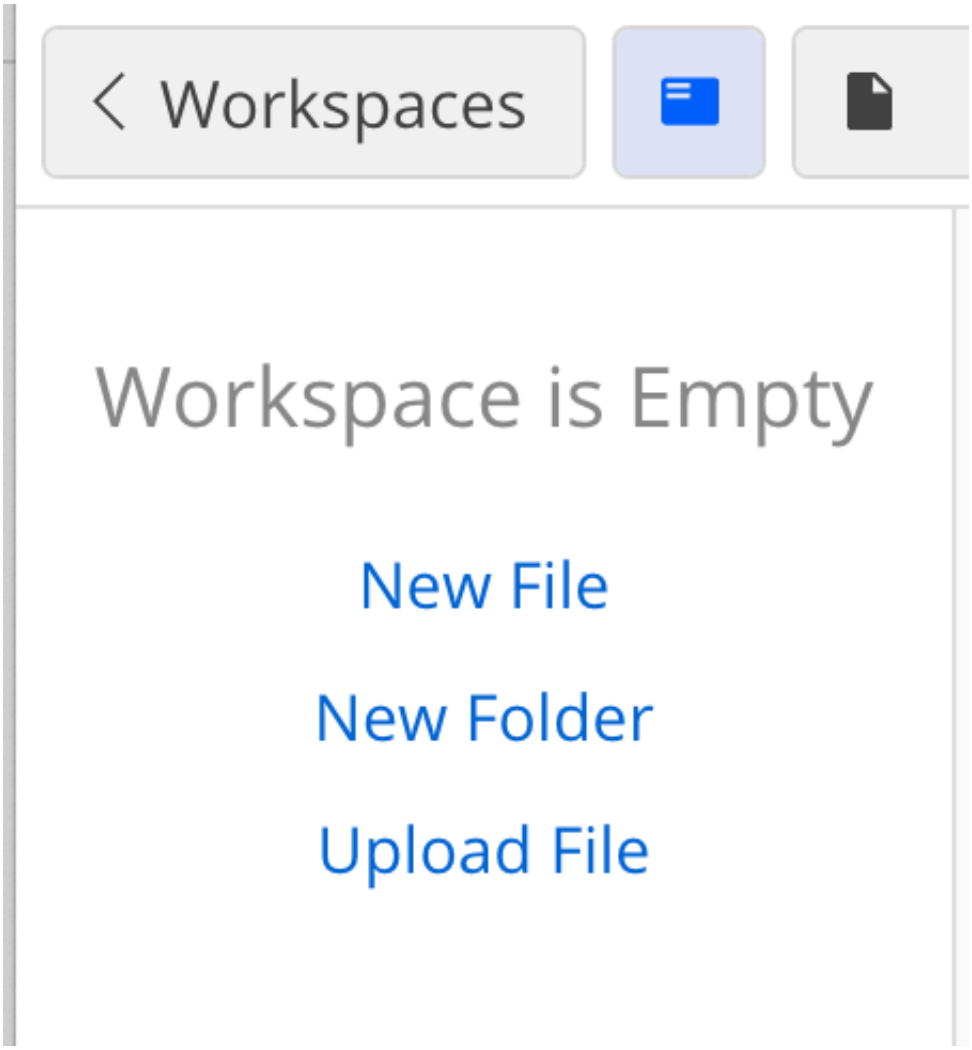
```
[user@sahara ~]$ python -i
Python 3.9.0 (default, Oct  7 2020, 23:09:01)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> number_of_tutors = 20
>>> number_of_tas = 4
>>> number_of_staff = number_of_tas + number_of_tutors
>>> number_of_staff
24
>>> exit()
[user@sahara ~]$
```

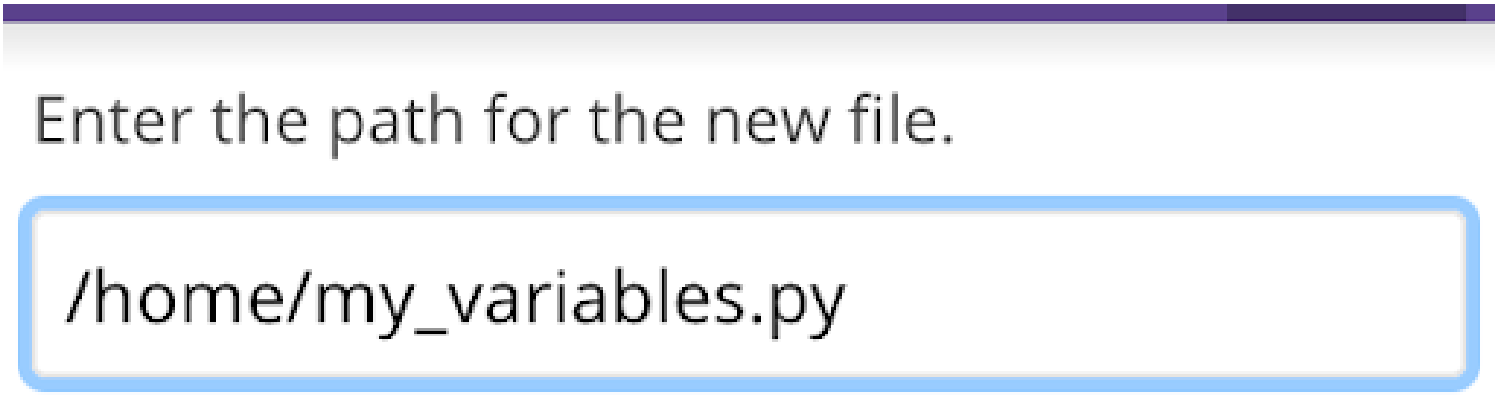
Quit your Python session using exit(), then start a new one with `python -i` again. Try using the variables you just defined in the exercise above (like `x` and `number_of_staff`). What happens? What is a lesson you learned from this?

Variables Definitions and Running a Python File

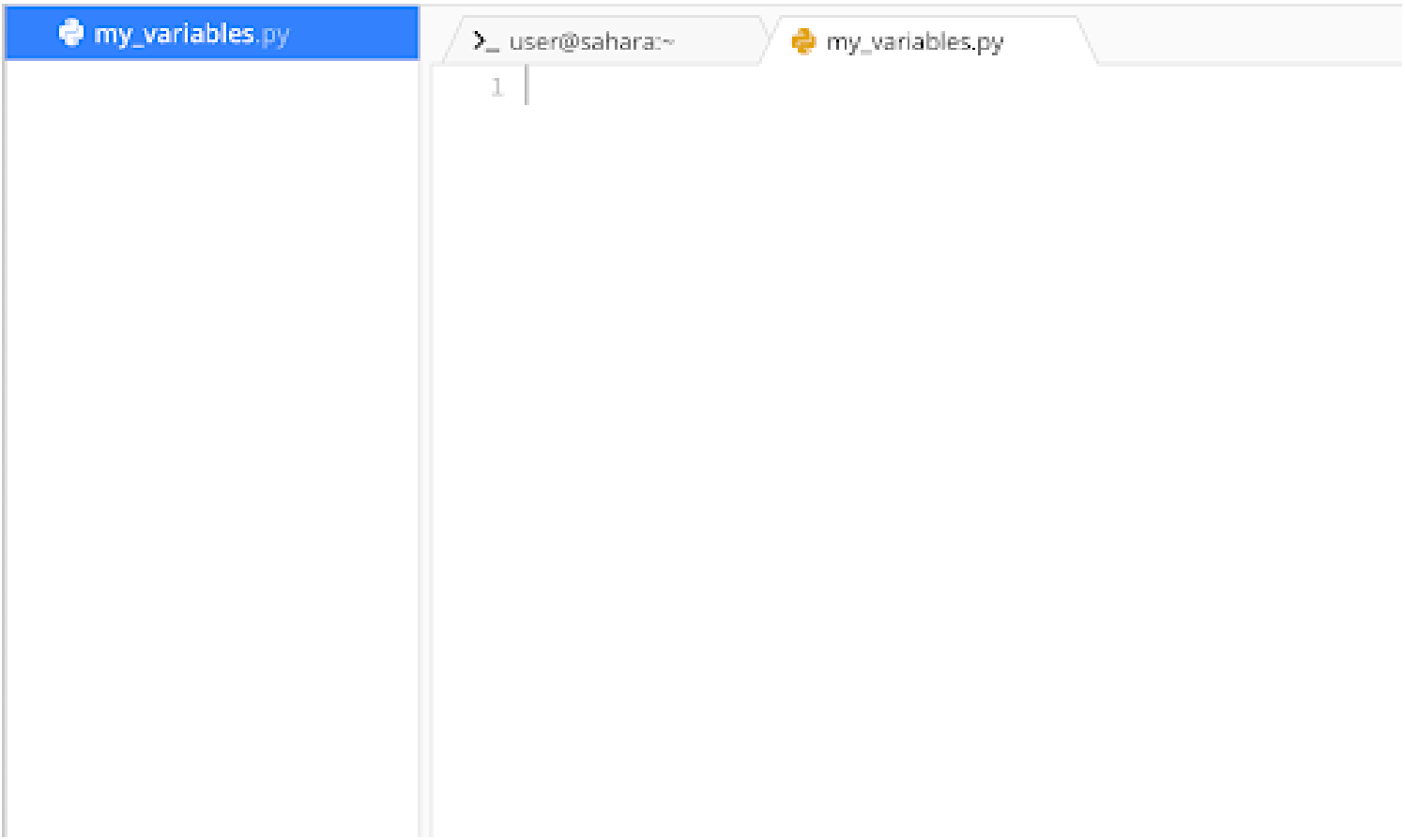
When we write programs, we usually end up wanting to save them in a file so that we can run the same program many times, rather than typing it into interactive Python every time.

In Ed, on the left hand side, click “New File,” and name the new file `my_variables.py`

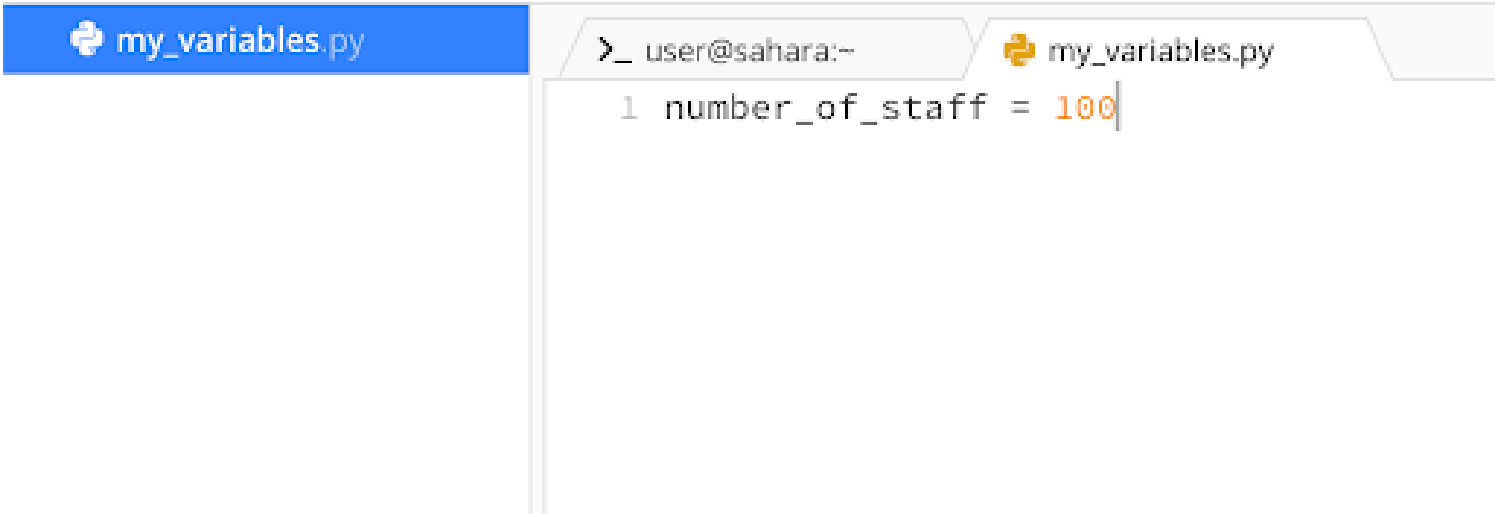




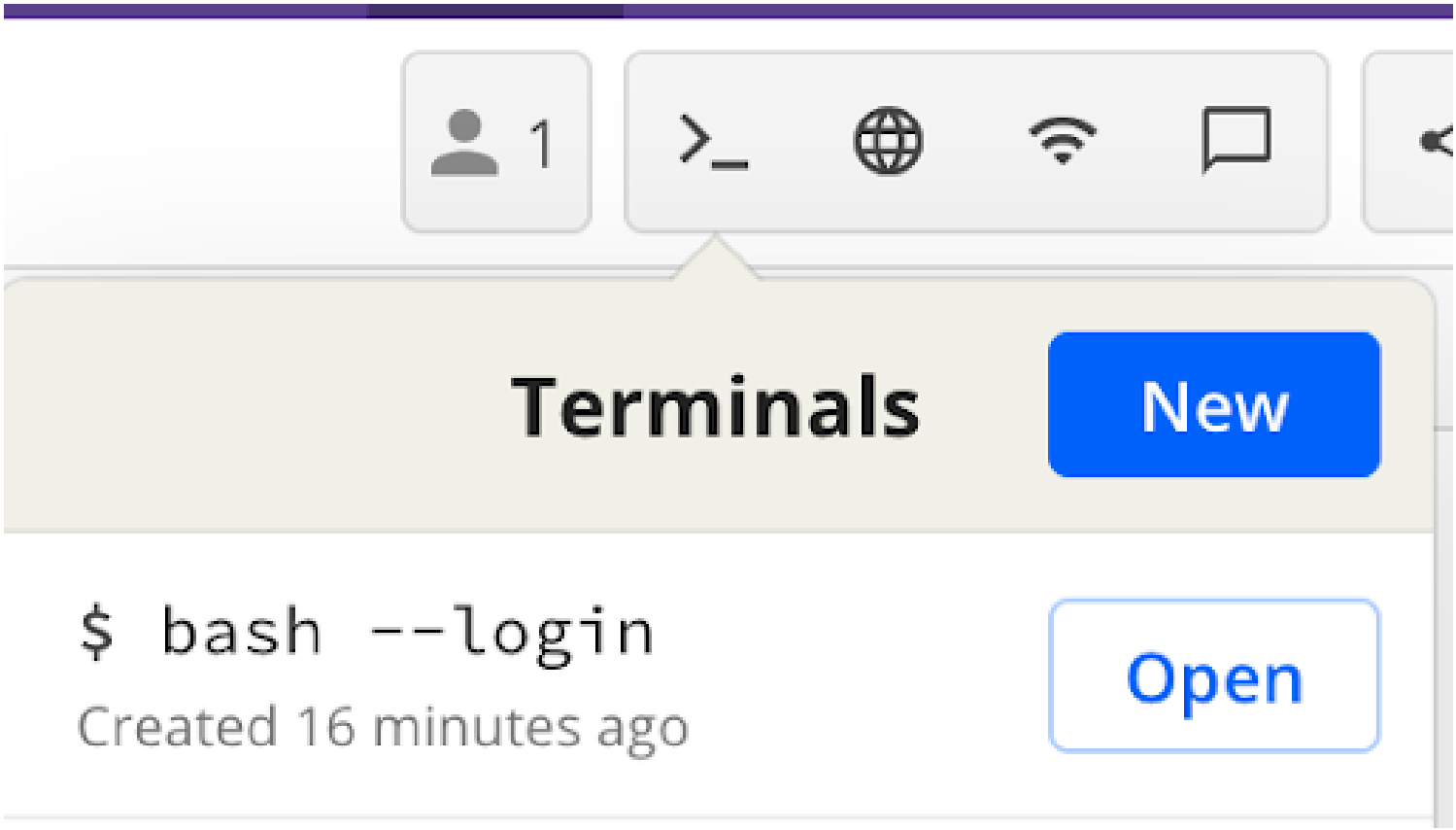
You should see a new tab open up:



We can write Python variable definitions (and other Python code that we'll see in the future) in this file, and then run it. Let's define a variable `number_of_staff` like we did before:



Now we can run the saved program (Ed automatically saves files when we edit them). Open a new terminal, which Ed will conveniently put at the bottom of the screen so you can see your program and the terminal at the same time:



In the new terminal, type: `python -i my_variables.py` and press Enter.

Now type in the name `number_of_staff` and press Enter. The definition from the file is now recognized by interactive Python:

Quit Python (remember how?) and then run just `python -i` (don't include the name of the file). Try using the variable `number_of_staff`. What happens? What does this tell you about Python?

Pick an experiment to try that involves starting Python and writing variable definitions in files. You might:

- Create a new file and try running that file with `python -i` to see what variables are defined
- Edit the file you already made to add new variable definitions and try running the file/using those variables to see their values
- Something else you think of that involves variables and files and `python -i`

Spend a few minutes on this. Feel free to share any questions you have.

Part 2: Quiz [15 mins]

Now that you have completed the lab, you will take a short mandatory quiz on the lab material. To take the quiz, tutors in your lab will provide you with the passcode to access the Lab Quiz on your Ed account. This will be a **timed (15 mins)**, multiple choice quiz on the lab material. If you completed the lab, you will be able to answer all of the questions on the quiz.

You are required to take the quiz **individually** during your lab session. Make sure to leave enough time to take the quiz before the end of the lab session.