# CIS 1100

Types, Variables & Sequences! (Lecture)

Python

Fall 2024

University of Pennsylvania

## Correction: Pritnting { and } in f-strings

If you want to print `{` in an fstring, you don't escape it with a `\`, insetad you "double it" within a `{}` pair

example:

```
print(f"{1}}}") # prints "1}"
```

Note: You are not expected to know this, and we will not test on this specific part of f-strings. I still want to correct something I said wrong last lecture.

## Review: Boolean Type

Another type that exists is `bool` which can represnt two values `True` or `False`

```
x = True
y = False
print(x)
```

can use the operators `and`, `or` and `not` on booleans

What does `c` evaluate to? (S7)

e.g. if we ran this code then printed `c`, what would it print?

```
a = False
b = True
c = (not a or b) and not (a and True)
```

## Review: Comparing

A common way to get boolean values is through comparison.

- `==` checks if two things are equal
- `!=` checks if two things are NOT equal

`"Hello" == "hello"` evaluates to False

`5 != 3` evaluates to True

`"hi" == "hi"` evaluates to True

## Review: Ordering

There also exist oeprators to check for:

- `<=`: less than or equal to
- `=>`: greater than or equal to
- `<`: less than
- `>`: greater than

## Review: Numerical Types

Python can store numbers, but it makes a distinction between two types of numbers:

- `int` These are Integers, meaning any positive or negative value (or zero).
  - e.g. `0`, `-3200`, `10`
- `float` These can store rational numbers and some special values
  - e.g. `3.14`, `2.0`, `infinity`

# Review: Other Arithmetic Operators

- `**` used for exponents.
  - e.g. 5 squared is `5 ** 2`
- `//` used for "integer division, rounds the result towards 0
  - `int // int` evaluates to an `int`
  - `3 // 2` evaluates to `1`
- `%` called "modulo" used to get the remainder of a division.
  - `5 % 2` evaluates to `1`
  - `9 % 3` evaluates to `0`

Write some code to determine if the variable x contains an even integer (L11)

```
x = <some_number>
is_even = _________________
```

## Opeartor Precedence

When we start combining operators together, it can get pretty complex to figure out the order things are evaluated.

Basic order of operations: (First evaluated to last evaluated)

- Parenthesis `()`
- exponents `**`
- multiplication/division/mod `*` `/` `//` `%`
- addition / subtaction `+` `-`
- comparisons and membership `in` `not in` `<` `>=` `==` etc.
- `not`, `and`, `or`

**You do not need to memorize these. When in doubt: Use Parenthesis**

## Type Conversion

Allows you to convert from one type to another

```
x = int("1300")    # converted str to int
z = str(x)         # Z has str conversion of x "1300"
a = bool("True")   # a has bool value True
f = float("3.14") # f has float value 3.14
```

## Complex Reassignment

What does `z` evaluate to after this code is run? (S8)

```
x = 3
y = "luv sic"
z = str(x > 0 and x <= 6) + " " + y + str(x)
```

Compressed Order of Operations

- PEMDAS
- comparison
- boolean operations (and/or/not)

## TypeError

Some operands cannot be done between some types

Consider:

```
x = 3 + "Howdy"
y = "bleh" > True
```

Most of the time you can resolve this by just adding a type conversion

## Invalid Casts

Not all types can always be converted from one to another.

Consider:

```
x = int("Howdy") # invalid literal for int() with base 10: 'howdy'
```

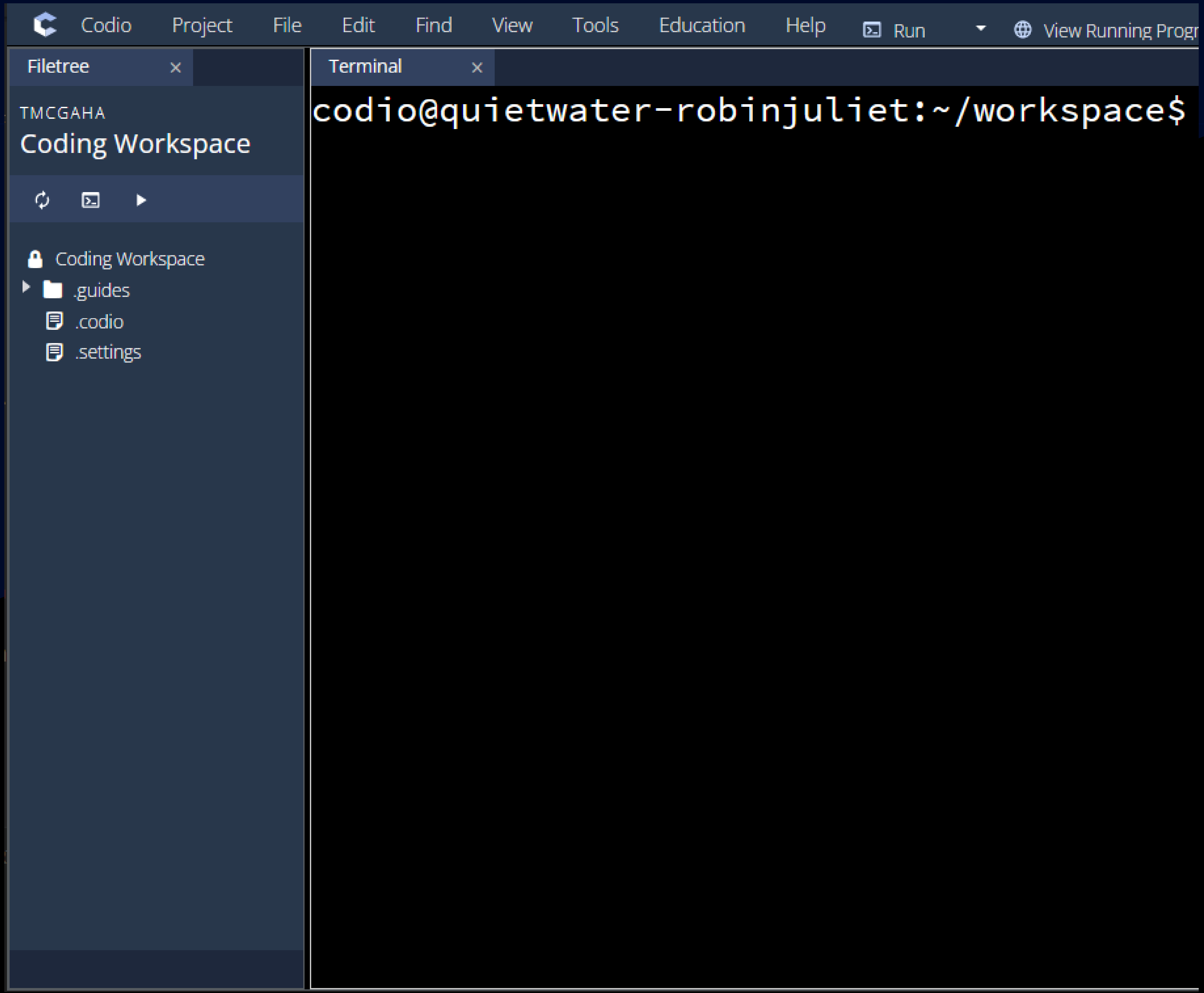
# Python REPL

Python has a REPL that you can use to run python code without storing it in a file.
REPL = **R**ead **E**valuate **P**rint **L**oop

Useful if you want to just check something real quick

May be useful to write a regular python file for more longer or complex code.

To run, you can go to the "Run" or "terminal" tab in codio and type the command `python`



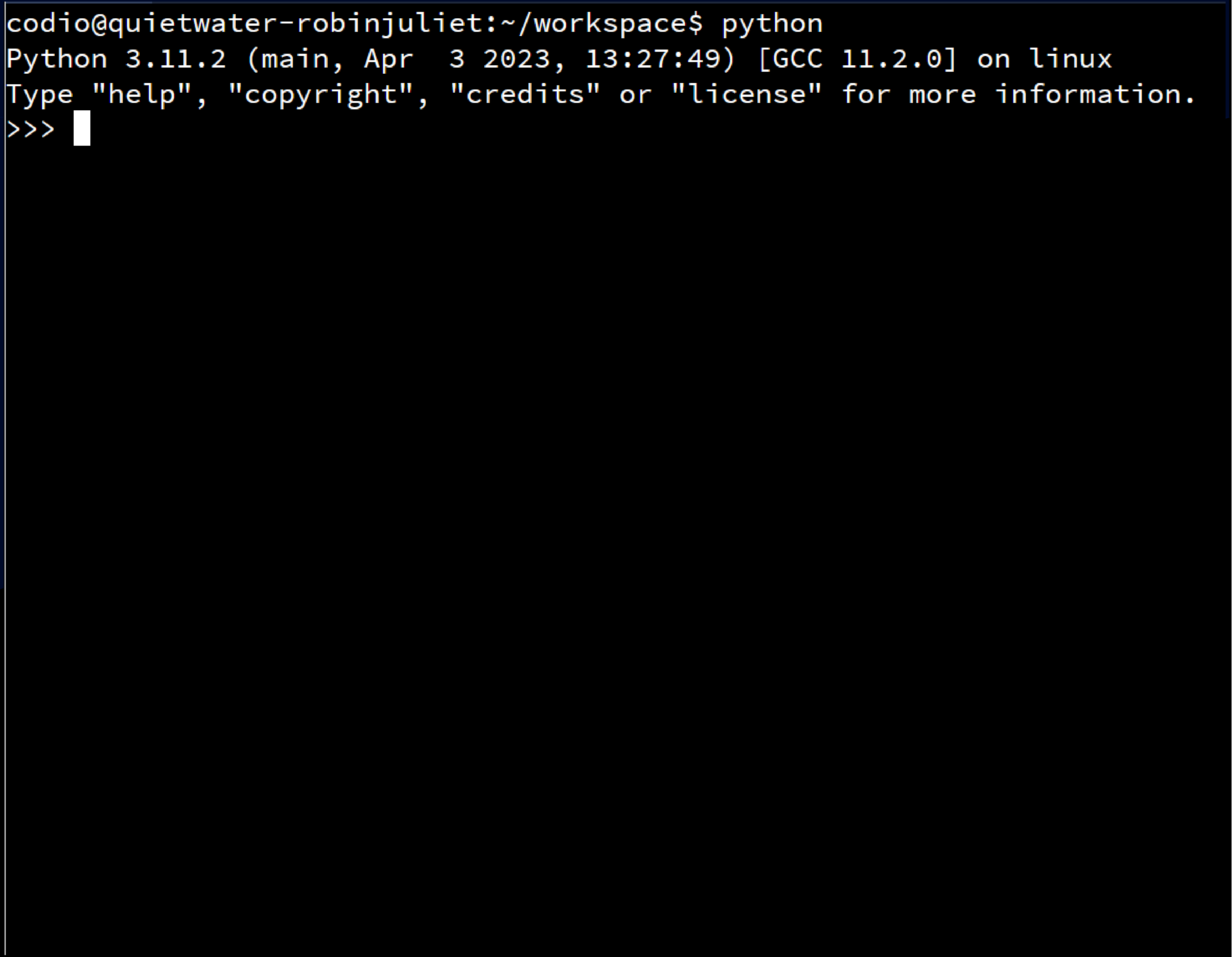
```
codio@quietwater-robinjuliet:~/workspace$
```

## Python REPL

To run, you can go to the "Run" or "terminal" tab in codio and type the command `python`

**Remember to end any running program first before you try to run the REPL**

Once you have it open, you should be able to see it prompt you in the terminal:



```
codio@quietwater-robinjuliet:~/workspace$ python
Python 3.11.2 (main, Apr  3 2023, 13:27:49) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

# Python REPL

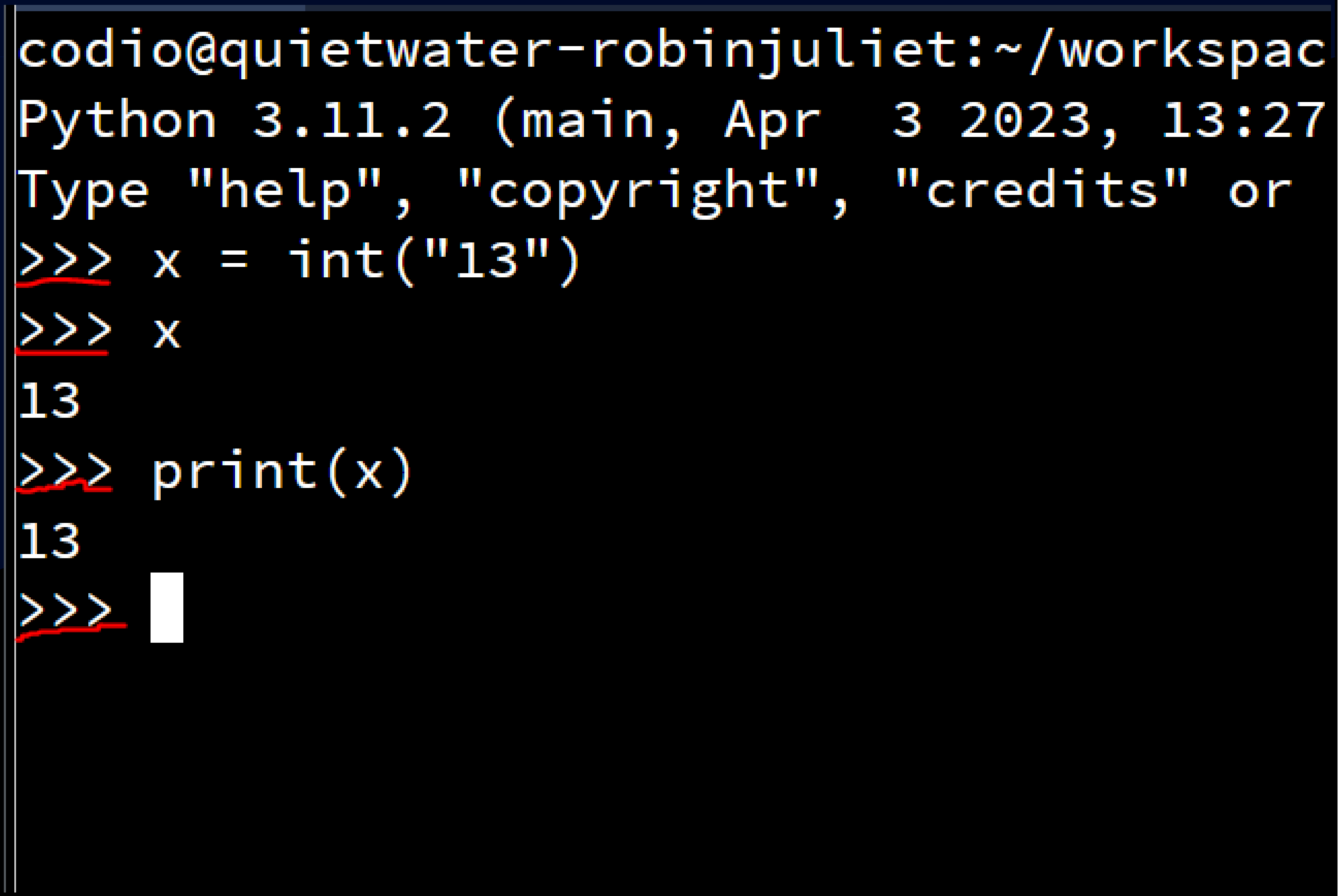
Once you have it open, you should be able to see it prompt you in the terminal:

I underlined the prompt in red

you can then type in lines of python and it will run those lines and remember variables.

if you don't assign a value into a variable, it will print that value.

Use `CTRL + D` or type `exit()` when done.



```
codio@quietwater-robinjuliet:~/workspac
Python 3.11.2 (main, Apr  3 2023, 13:27
Type "help", "copyright", "credits" or
>>> x = int("13")
>>> x
13
>>> print(x)
13
>>> 
```

## Sequence Types: String

An important aspect of the string type, is that it is a sequence type. A string contains a *sequence* of characters.

Sequences have a length and indexes for individual members of the sequences.

For strings, it is a sequence of characters. Emphasis on sequences because: **The order matters**

```
s1 = "hi"
s2 = "ih"
print(s1 == s2) # False
```

## Sequence Types: String

Ordering is an important aspect to string, and we use **indexes** as a way to specify positions in the string.

Consider the string

| **index** | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| characters | H | e | l | l | o | ! |

index `0` is the first position (first character)

`len(string) - 1` is the index of the last character

## Sequence Types: Basic Functionality

For any sequence, you can use

- `len(seq)` to find the length of the sequence `seq`
- `seq[i]` to access the `i`th index of the sequence `seq`

```
x = "Travis"
print(x[0])          # 'T'
print(x[3])          # 'v'
print(x[len(x) - 1]) # 's'
```

## Practice:

What is the index of the letter `E` and `Y` from the following string (S9)

```
x = "GY! BE"
```

Get the last character of a string. Your code should work for any value of string that has len >= 1. Do not just say `last_char = "e"`. (S10)

```
string = "example"
last_char = ________
```

## Practice:

Get the middle character of a string withouth knowing the string's value. Assume length is odd and >= 1. (C12)

```
string = "example"
middle_char = "
```

Hint: `//`

Get the age from this string an int. Do not just say `age = 26`, actually extract it from the string. (Also C12)

```
string = "Age: 26"
age = ___________________
```

Hint: assume age is two characters. Think about what operators are useful here.

## Other String Sequence Function

- `string.find(target)`: finds the first index that has the target string, or -1 if not found.

example:

```
"Hello".find("l")  # 2
"Phl".find("U")    # -1
"Attack".find("ta") # 2
```

## Reminder:

- There is another check-in due before lecture as always.
  - Friday's check-in will have an "exit-ticket" for you to submit questions and metrics about the course.
- Office Hours and Recitation start this week!
  - Recitation counts attendance, show up to your assigned recitaiton!
- HW00 is out and due wednesday (9/11) at midnight
- Expect HW01 to be released sometime this week