Objects & Data Types

Announcements

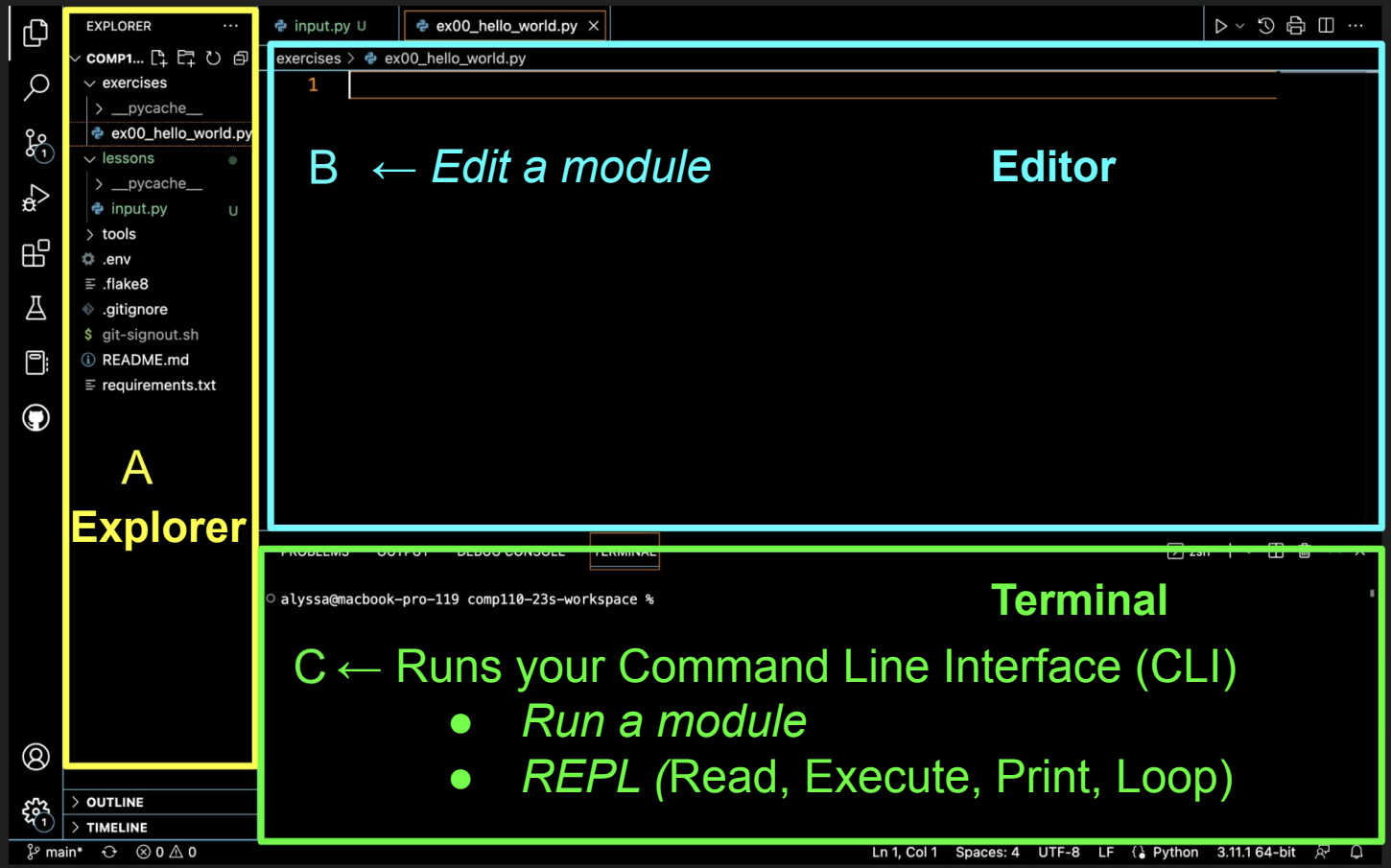
- Want help installing software, setting up your workspace, or practicing today's content?
  - Visit our Open House today from 11am-5pm in Sitterson Hall, room 008!

- Homework:
  - Set up your course workspace
  - **LS01: VS Code, Terminal, + Running Programs** due tonight at 11:59pm
  - **LS02: Objects and Data Types** due tonight at 11:59pm
  - **EX00: Hello World!** due Monday, August 25 at 11:59pm

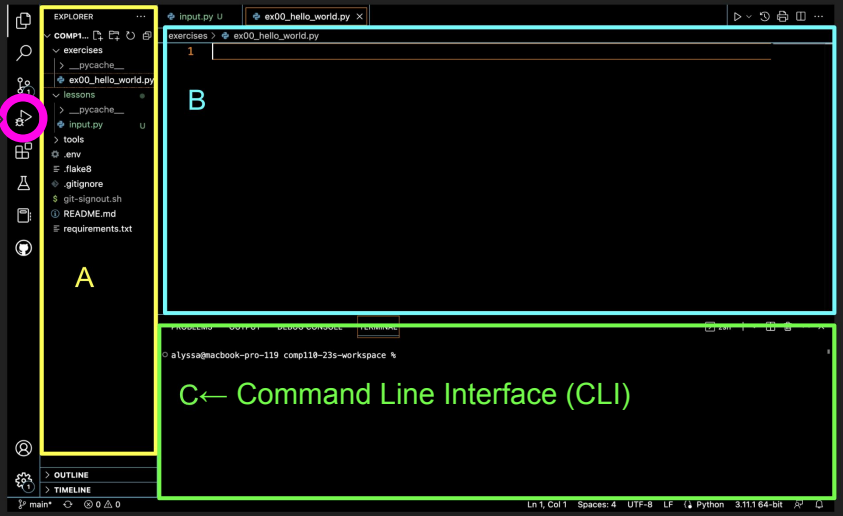
Office Hours begin tomorrow (Aug 21)

- Hours:
  - Mondays-Fridays: 11am-5pm in SN008
  - Sundays: 1-5pm in SN008
- We use the **CS Experience Labs (CSXL) website**
- General Rules:
  - Must submit a ticket to be seen
  - Limited to 15 minutes and one specific question per appointment
- Completely lost? *Try tutoring!*
  - Best for longer-form help (> 15 mins) and conceptual questions
  - Hours and locations will be shared soon!

An introduction to Visual Studio…

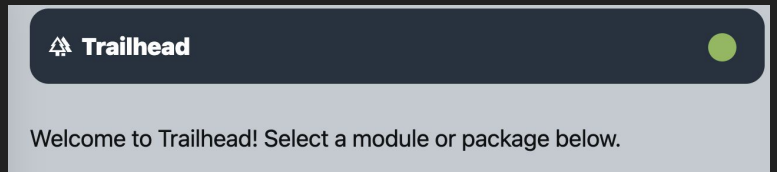


Ways to run code



Use Trailhead:

- Launch with the debug button
- “Starting Trailhead server at http://localhost:1110”



Interactive (like a conversation with your computer):

REPL: Read Execute Print Loop

- To initialize the REPL in your terminal, type:
  - `python`
- `>>>` means you’re in the REPL

To run a module (execute a python (.py) file) from your terminal, type:

- `python -m my_file_name`

Some lines of code exist exclusively for human readability!

Docstrings

- A string written at the top of every file to describe its purpose
  - Written for humans, not for the computer to evaluate
- Denoted with three quotations “““ ”””

Comments

- Lines that start with # are ignored by the interpreter
  - Written for humans, not for the computer to evaluate
- Best practice to comment your code to explain what it’s doing (if it isn’t obvious!)

Objects and Built-In Types

An **object** is *typed* unit of data in memory.

The object's **type** classifies it to help the computer know how it should be interpreted and represented.

Programming languages offer many built-in data types for you to work with, typically including:

- numerical
  - integers
  - decimal numbers
  - complex numbers
- textual
- logical
- collections of many objects
  - sequences
  - sets
  - dictionaries

Numerical Built-In Types

- Integers
  - `int`
  - Zero, *or* non-zero digit followed by zero or more integers
  - 100 is an int, but 0100 is not
  - 3 is, but 3.08 is not
  - -2000 is, but -2000.1 is not
- Floating-point “decimal” numbers
  - `float`
  - Examples: 3.02, 4008.0, -16.99999
  - Not the *only* way to represent decimal numbers, but a very precise way

Boolean Built-In Type

- `bool`
- Evaluates to `True` or `False`
- Important: these should *always* have a capital T or F!
  - `True` is a boolean value
    - `TRUE` and `true` are not
  - `False` is a boolean value
    - `FALSE` and `false` are not

Textual Built-In Type

- Strings
  - `str`
  - A sequence (or *string*) of characters
  - Can be denoted using “ ”
  - Examples:
    - A word: “hello”
    - A phrase: “Hope we get some snow!”
    - A single character: “A”, “ “, “🎉”
    - A number *in quotes*: “23”, “110”, “12.5”
    - An empty string: “”

Indexing

- **Subscription** syntax uses square brackets and allows you to access an item in a sequence
- **Index numbering starts from 0 (in Python)**

Example:

The string, “`happy`”

Indexing

- **Subscription** syntax uses square brackets and allows you to access an item in a sequence
- **Index numbering starts from 0 (in Python)**

Example:

The string, "`happy`"

Characters:	`h`	`a`	`p`	`p`	`y`
Indices:	`0`	`1`	`2`	`3`	`4`

"`happy`"`[0]` would give us what letter?

In English: "happy at index 0"

Indexing

- **Subscription** syntax uses square brackets and allows you to access an item in a sequence
- **Index numbering starts from 0 (in Python)**

Example:

The string, "`happy`"

Characters: `h a p p y`

Indices: `0 1 2 3 4`

`"happy"[0]`? `"h"`

Your turn: `"happy"[4]`?

In English: "happy at index 0"

Check an Object's Type

- `type()`
  - `type(3)`
  - `type("hi")`
  - `type(True)`

Change (Cast) an Object's Type

- `float(3)`
- `str(3)`
- `int("3")`

Literals

A literal in Python **is a syntax that used to completely express a fixed value of a specific data type, in the simplest way possible**.

Examples:

- 4
- “hello”
- 3.8
- False

`str(3)` is not a literal; it is not the simplest way to express the string, `“3”`

Review: Data Types

Discuss these questions with your neighbor and jot the answers down.

1. What is the difference between `int` and `float`?

2. Is there a difference between the following? What *type* of **literal** is each an example of?
   a. `"True"`
   b. `True`
   c. `TRUE`

3. What role do **types** play for data in Python?

Review: `str` is a *Sequence* Type

Discuss these questions with your neighbor and jot the answers down.

1. What does the `len()` function evaluate to when applied to a `str` value? What will the expression `len("cold")` evaluate to?

2. Is there a difference between `"True"` and `'True'`? What *type* of **literal** is each an example of?

3. What are the **square brackets** called in the following *expression*? What does the following expression evaluate to? `"The Bear"[4]`

4. Can a string be a number in Python? Explain.