modules!

Organizing a Project into Packages and Modules

- As programs grow, you will organize them into **packages** and **modules**
  - In Python, a package is a directory, and a module is a Python file
  - We will only cover the fundamentals, for a more complete story: https://docs.python.org/3/tutorial/modules.html

- Global names in modules are *importable* in other modules
  - Environment diagram connections:
  1. This is every name bound in the Globals frame!
  2. Names include both function names and global variables/constants.

- Package/Module Paths follow directory structure with dot delimiters:
  - Directory Path: `lessons > ls24_modules.py`
  - Package Path: `lessons`
  - Module Path: `lessons.ls24_module`

Importing Specific Names from a Module

- To import names directly from a module:

```
from [module] import [global name_0], ..., [global name_N]
```

- Suppose `ls24_module` defined a global function named `total`:

```
def total(input: List[float]) -> float:
    # Elided
```

- Example - To import `total` from another module:

```
from lessons.ls24_module import total
```

- Imported names a *bound* to the same definitions they were bound to in the ***from*** module.

Importing an Entire Module

- To import an entire module:

```
from [package] import [module name]
```

- After importing a module, you can reference its global names with the following form:

```
[module name].[global name]
```

- Continuing from the previous slide's example:

```
from lessons import ls24_module
```

- After doing so, you could call its total function in the following way:

```
ls24_module.total([1.0, 2.0, 3.0])  # Returns 6
```

- This is generally a better practice than importing names directly once you are comfortable with it.
  - Why? It gives you access to *all* of a module's functions without introducing a lot of extra names into your module.

The Import Process

- When importing from a module, the entire module gets evaluated
  - Even if you're importing a single name!

- When you import a module, a special global variable `__name__` is a string containing the module's path.
  - In the previous example: `"lessons.ls24_module"`

- When you run a Python file *as a module* using the `-m` option, the global variable `__name__` is set to `"__main__"`.

- The idiomatic way to write a Python module that is both *"runnable"* and its names are easily importable is to add at the end:

```
if __name__ == "__main__":
    main()
```