

---

# RELATIVE IMPORTING PYTHON

---

```
Parent_folder/  
└─ main.py  
   └─ mods/  
      ├── __init__.py  
      ├── perimeter.py  
      └── point.py
```

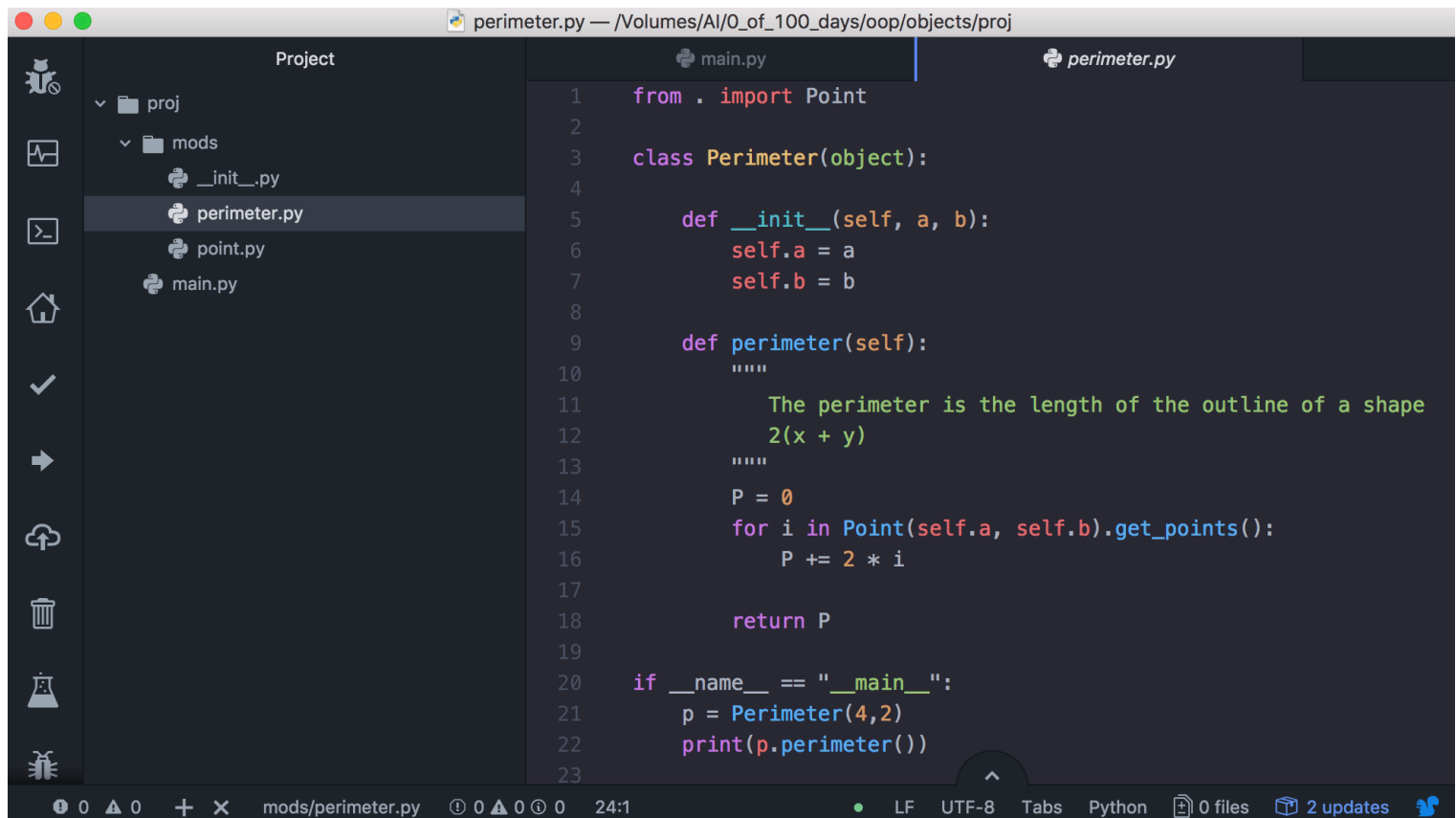
1 directory, 4 files

## Packages in Python

How to wrap a project as a package with modules that can perform relative importing. This is an example of such a project structure.

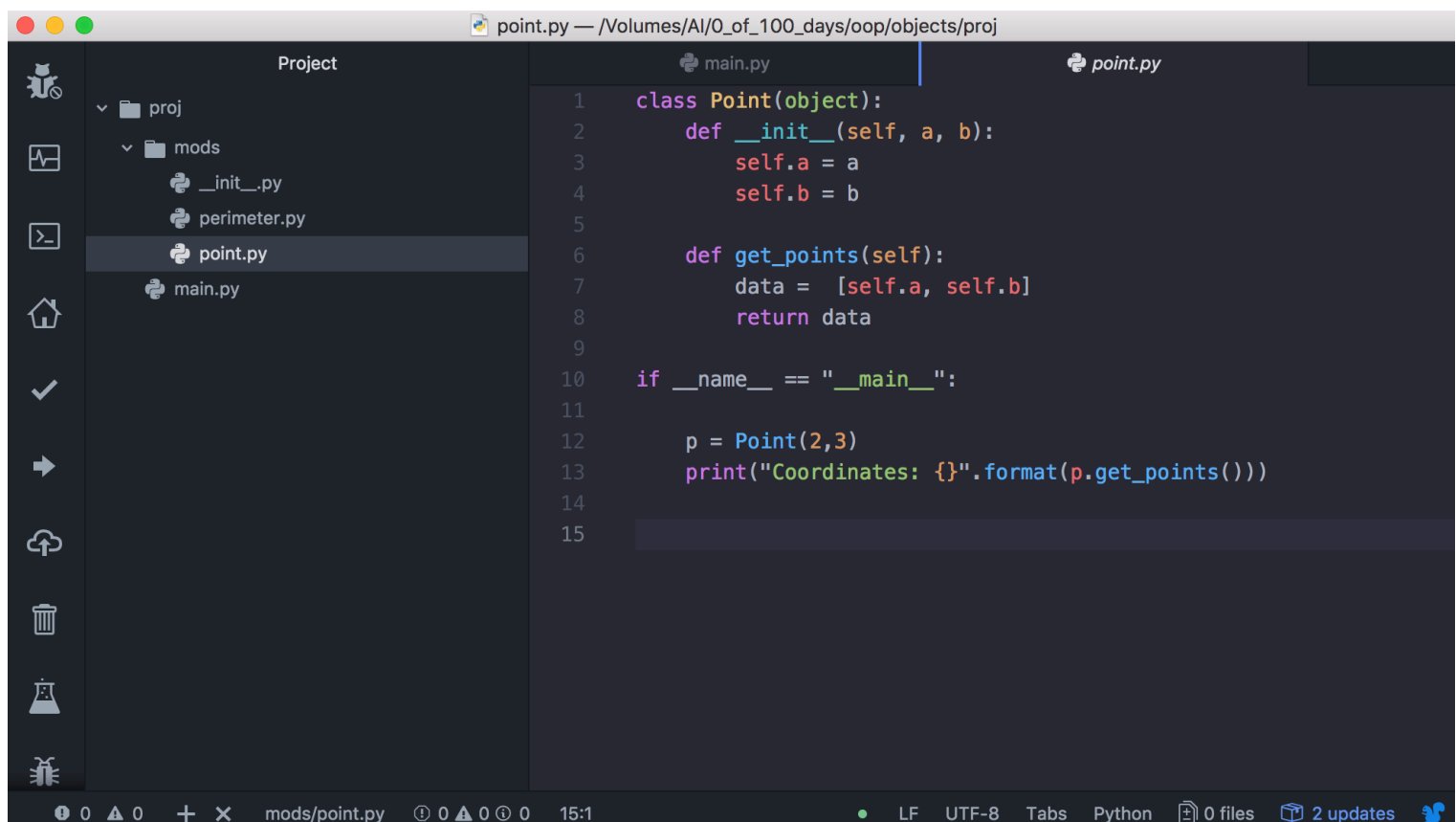
# Modules, \_\_init\_\_.py and main.py

perimeter.py

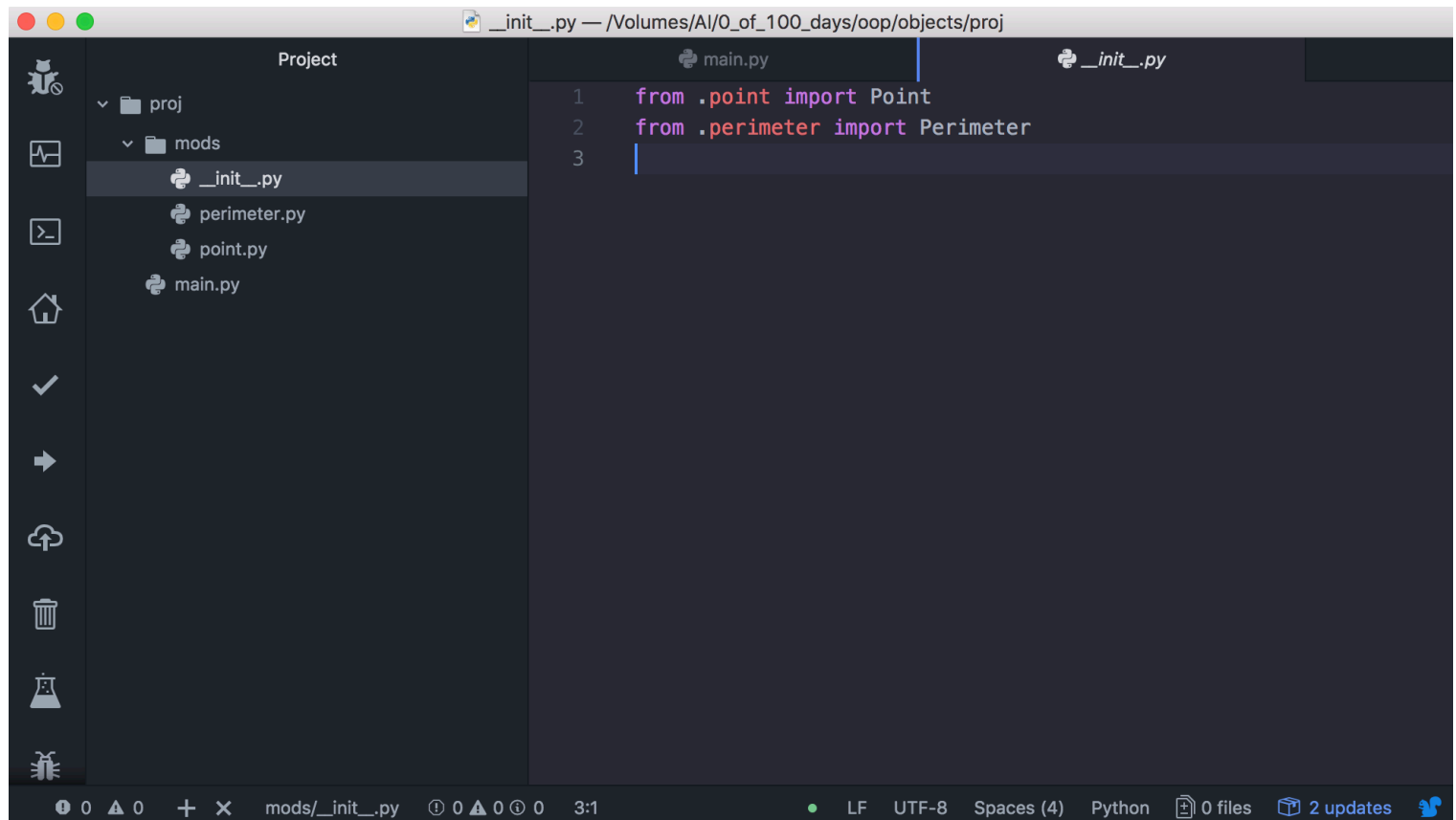


```
1 from . import Point
2
3 class Perimeter(object):
4
5     def __init__(self, a, b):
6         self.a = a
7         self.b = b
8
9     def perimeter(self):
10        """
11        The perimeter is the length of the outline of a shape
12        2(x + y)
13        """
14        P = 0
15        for i in Point(self.a, self.b).get_points():
16            P += 2 * i
17
18        return P
19
20 if __name__ == "__main__":
21     p = Perimeter(4,2)
22     print(p.perimeter())
23
```

point.py

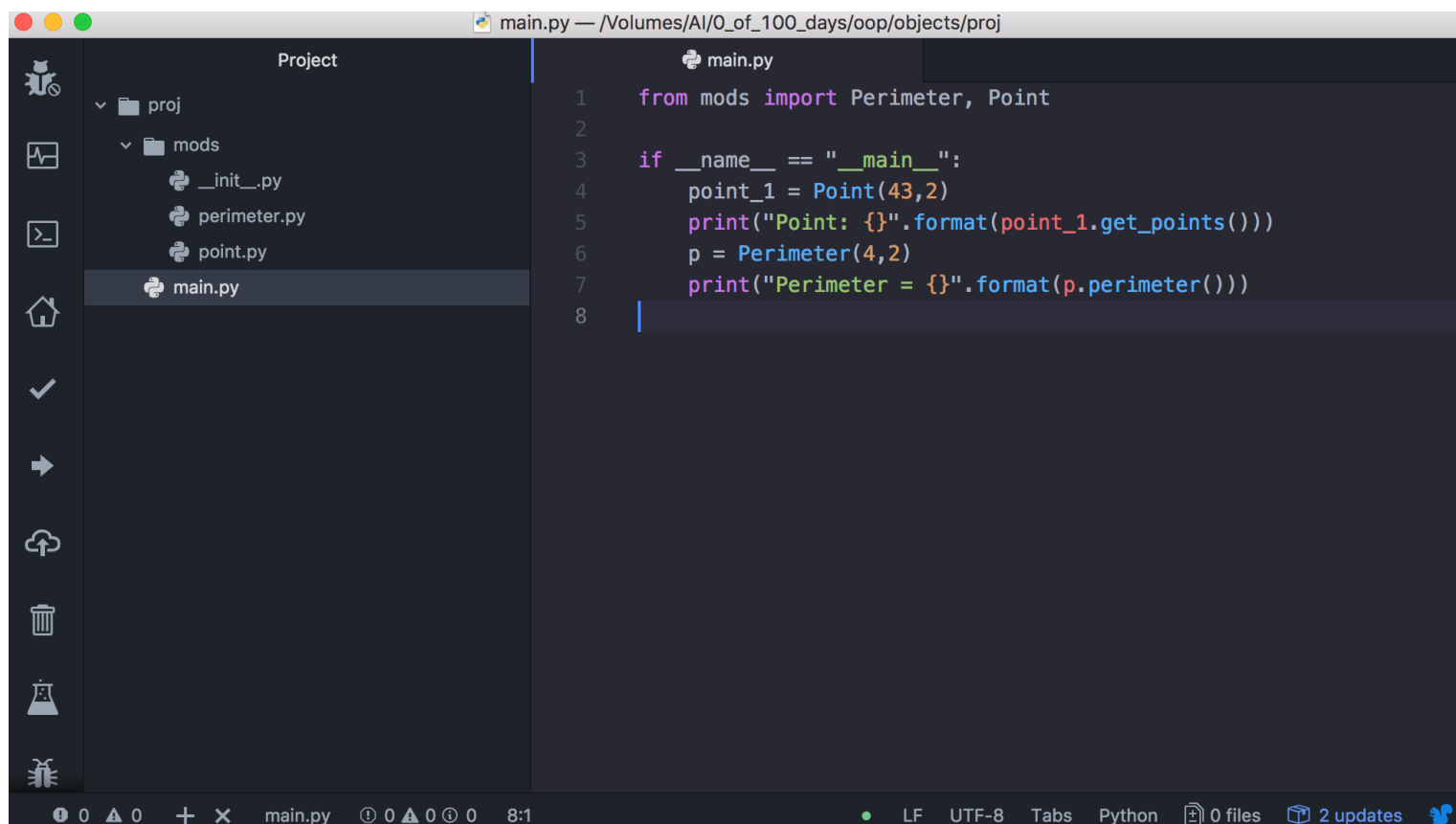


```
1 class Point(object):
2     def __init__(self, a, b):
3         self.a = a
4         self.b = b
5
6     def get_points(self):
7         data = [self.a, self.b]
8         return data
9
10 if __name__ == "__main__":
11
12     p = Point(2,3)
13     print("Coordinates: {}".format(p.get_points()))
14
15
```

`__init__.py`

The screenshot shows an IDE window titled `__init__.py — /Volumes/AI/0_of_100_days/oop/objects/proj`. The left sidebar displays a project tree with a folder `proj` containing a subfolder `mods`. Inside `mods`, the files `__init__.py`, `perimeter.py`, `point.py`, and `main.py` are listed. The `__init__.py` file is selected. The main editor area shows the content of `__init__.py`, which contains two import statements: `from .point import Point` and `from .perimeter import Perimeter`. The status bar at the bottom indicates the file is in `mods/__init__.py` with a line number of 3:1, using LF line endings, UTF-8 encoding, 4 spaces for indentation, and is a Python file with 0 files and 2 updates.

```
1 from .point import Point
2 from .perimeter import Perimeter
3
```

`main.py`

The screenshot shows the same IDE window, but now the `main.py` file is selected in the project tree and the editor. The window title is `main.py — /Volumes/AI/0_of_100_days/oop/objects/proj`. The editor displays the content of `main.py`, which imports `Perimeter` and `Point` from the `mods` package. It then creates a `Point` object, prints its points, creates a `Perimeter` object, and prints its perimeter. The status bar at the bottom indicates the file is in `main.py` with a line number of 8:1, using LF line endings, UTF-8 encoding, and is a Python file with 0 files and 2 updates.

```
1 from mods import Perimeter, Point
2
3 if __name__ == "__main__":
4     point_1 = Point(43,2)
5     print("Point: {}".format(point_1.get_points()))
6     p = Perimeter(4,2)
7     print("Perimeter = {}".format(p.perimeter()))
8
```