Implementing Packages

Austin Bingham
COFOUNDER - SIXTY NORTH
@austin_bingham   sixty-north.com

Overview

Define your own packages

Understand package initialization

Relative imports

Control default package imports

Creating Packages

PEP420 and `__init__.py`

`__init__.py` optional in Python 3.3+

Still required in earlier Python versions

Powerful initialization tool

“Explicit is better than implicit”

A package is a directory containing `__init__.py`

Project: `MultiReader`

Read uncompressed text files

Read gzip-compressed files

Read bz2-compressed files

Creating a Subpackage

```python
import gzip
import sys

opener = gzip.open


if __name__ == '__main__':
    f = gzip.open(
            sys.argv[1],
            mode='wt')
    f.write(' '.join(
        sys.argv[2:]))
    f.close()
```

◀ Alias for `gzip.open`
  - Decompresses during read

◀ “main block”

◀ Use `gzip` to create compressed file

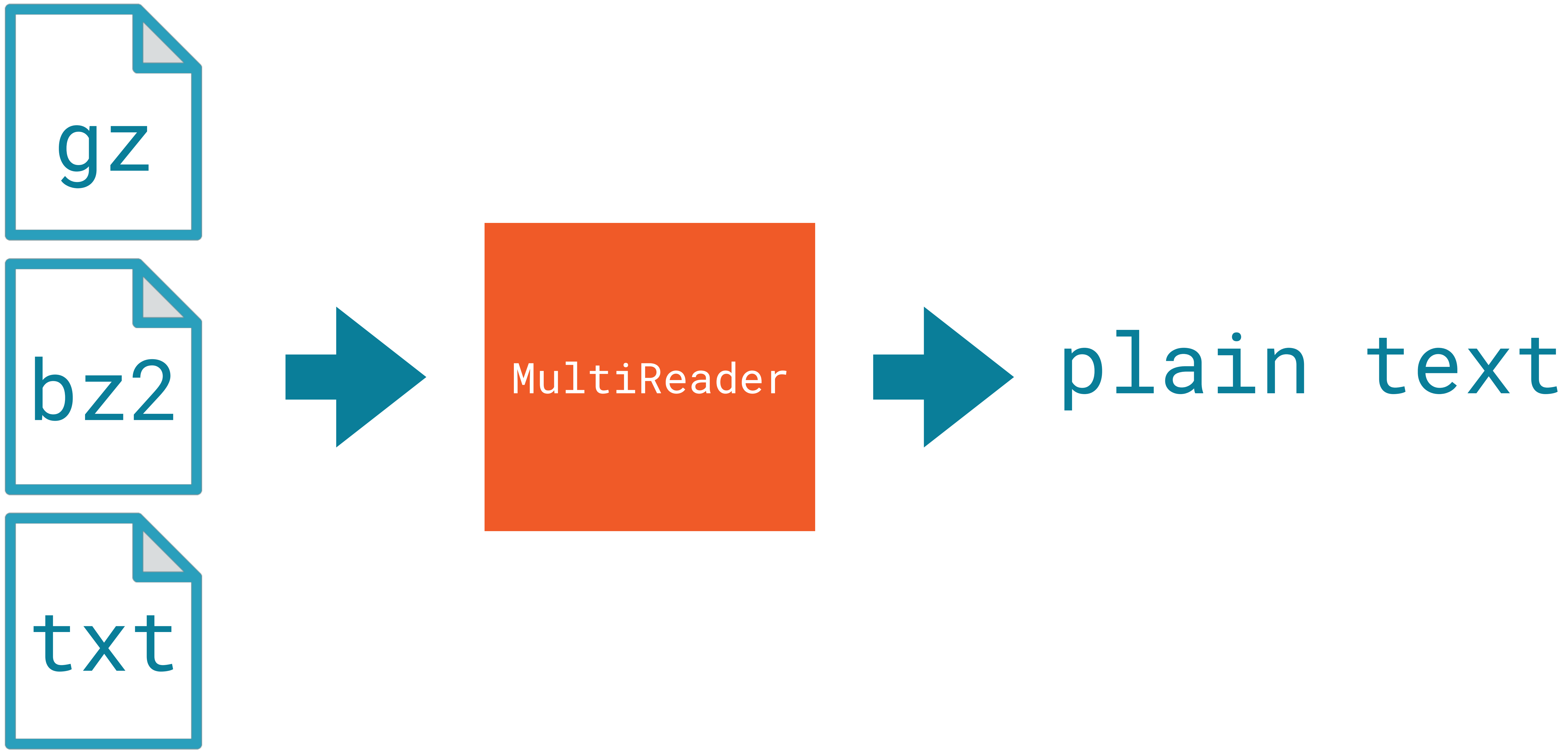
◀ Path to new compressed file

◀ Join to space-separated string

◀ The data to compress

```
demo_reader
  ├── __init__.py
  ├── multireader.py
  └── compressed
        ├── __init__.py
        ├── bzipped.py
        └── gzipped.py
```

MultiReader Program



Key changes to `MultiReader`

Checks for file extension in `extension_map`

If found, specialized opener is used

By default `open()` **is used**

Relative Imports

Absolute Imports

```
# Both of these absolute imports mention both ``demo_reader`` and ``compressed``

import demo_reader.compressed.bzipped
from demo_reader.compressed import bzipped
```

Relative Imports Syntax

```
from ..module_name import name
```

Parent of current module

Grandparent of current module

Important Rules for Relative Imports

You can only use the

`from module import name`

form of import

Relative imports can only be used to import modules within the current top-level package

Relative Imports from demo_reader/compressed/bzipped.py

Relative	Absolute
`from . import name`	`from demo_reader.compressed import name`
`from .. import name`	`from demo_reader import name`
`from ..util import name`	`from demo_reader.util import name`

Summary of Relative Imports

Can reduce typing in deeply nested package structures

Promote a certain form of modifiability

In general, prefer absolute imports

`__all__`

Module-level attribute

Controls `from module import *` behavior

If not specified, import all public names

Must be a list of strings

- Each entry is a name to import

While `__all__`
can be useful…

…we recommend avoiding
`import *` in general

Summary

Packages are modules which can contain other modules

Directories containing `__init__.py`

`__init__.py` **is...**

- Technically optional
- An explicit signal to developers
- Executed at package import

Packages can contain subpackages

`__all__` **controls** `import *` **behavior**