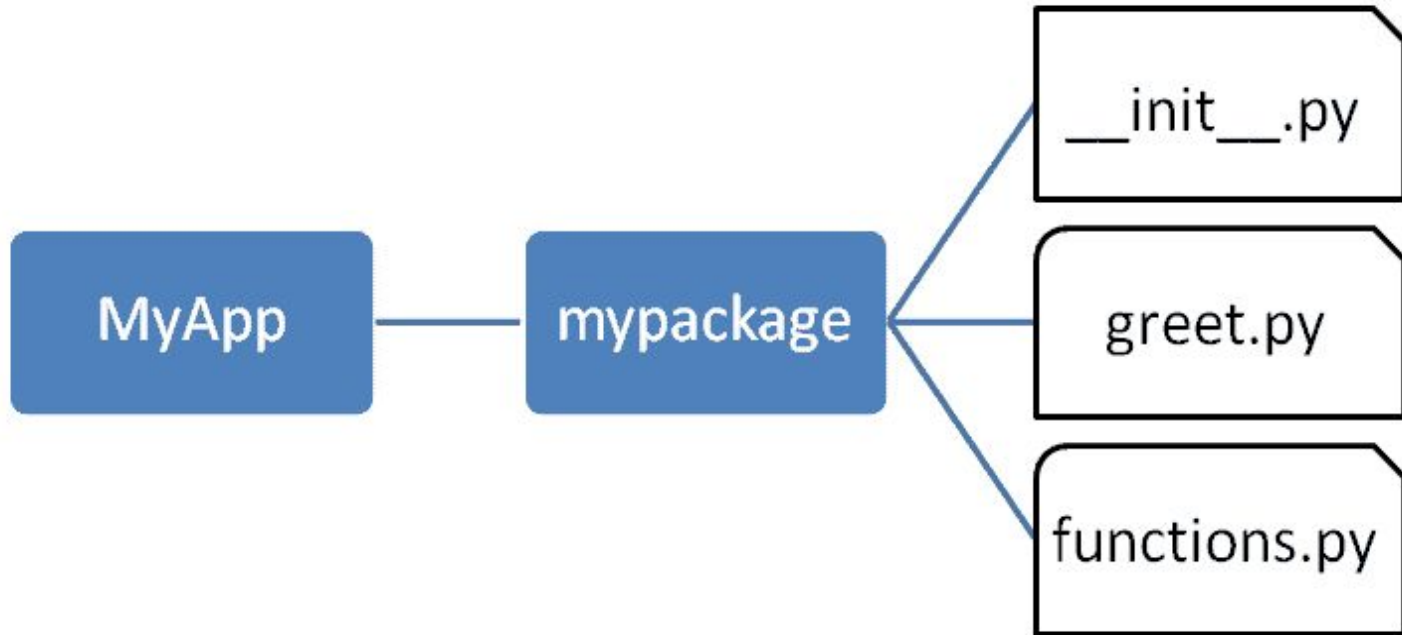


- **Package Discussion**
- **List Comprehension**
- **Dict Comprehension**



# Package in Python

- Package is a folder or directory.
- Collection of modules and packages.



Package Folder Structure

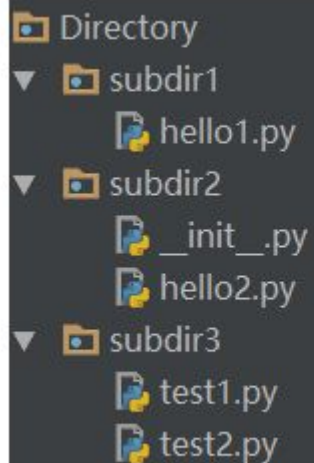


Regular Package ( With `__init__.py`)

Namespace Package (Without `__init__.py`)

- Before 3.3 `__init__.py` file was needed to add in any directory to create a package
- From python 3.3+ , namespace packages were introduced.
- Which does not need a init file.
- And gives flexibility to have sub\_packages on different directories.

# Example



```
Directory
├── subdir1
│   └── hello1.py
├── subdir2
│   ├── __init__.py
│   └── hello2.py
└── subdir3
    ├── test1.py
    └── test2.py
```

```
# test1.py
from subdir1 import hello1
hello1.hello()
```

```
# test2.py
from subdir2 import hello2
hello2.hello()
```

# List Comprehension



- **List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.**

**Example:**

- **Based on a list of fruits, you want a new list, containing only the fruits with the letter "a" in the name.**
- **Without list comprehension you will have to write a **for** statement with a conditional test inside:**

# Example

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
newlist = []
```

```
for x in fruits:  
    if "a" in x:  
        newlist.append(x)
```

```
print(newlist)
```

## Example 2 using List Comprehension

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
  
newlist = [x for x in fruits if "a" in x]  
  
print(newlist)
```

## Using Dictionary Comprehension

From the above example, we can see that dictionary comprehension should be written in a specific pattern.

The minimal syntax for dictionary comprehension is:

```
dictionary = {key: value for vars in iterable}
```

Let's compare this syntax with dictionary comprehension from the above example.

The diagram illustrates the mapping between the general syntax and a specific example of dictionary comprehension. The general syntax is shown at the top, and the specific example is shown at the bottom. Vertical lines connect the components of the general syntax to their corresponding parts in the example.

```
{ key: value for vars in iterable }
```

```
{ num: num*num for num in range(1, 11) }
```

The mapping is as follows:

- `key` maps to `num`
- `value` maps to `num*num`
- `vars` maps to `num`
- `iterable` maps to `range(1, 11)`