

Packages (1A)

Copyright (c) 2024 - 2015 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Modules and Packages

A **module** in Python is a single file that contains Python code in the form of functions, executable statements, variables, and classes.

A module acts as a self-contained unit of code that can be imported and used in other programs or modules.

A **package**, on the other hand, is a collection of modules organized in a directory.

Packages allow us to group multiple related modules together under a common namespace, making it easier to organize and structure our code base.

<https://www.sitepoint.com/python-modules-packages/>

Working with Modules

Modules can be imported and used in other programs, modules, and packages.

They're very beneficial in an application, since they break down the application function into smaller, manageable, and logical units.

We can put all the code in one file, but then the code very quickly becomes unmaintainable and unreadable.

By using modules, we can break down the code into units that are more manageable.

Breaking the code down into those modules promotes organization, reusability, and maintainability.

<https://www.sitepoint.com/python-modules-packages/>

Creating a simple module

have a number of related functions, variables, and classes:
we could put them in one **module**, and give the module any name we want

To create a **module** in Python, open up an IDE or text editor,
create a file, and give it a descriptive name and a .py extension.

For this example, let's call it **sample.py** and enter in the following code:

```
# create a variable in the module
sample_variable = "This is a string variable in the sample.py module"

# A function in the module
def say_hello(name):
    return f"Hello, {name} welcome to this simple module."

# This is another function in the module
def add(a, b):
    return f"The sum of {a} + {b} is = {a+b}"

print(sample_variable)
print(say_hello("kabaki"))
print(add(2, 3))
```

<https://www.tutorialspoint.com/python-modules-packages/>

Creating a simple module

have a number of related functions, variables, and classes:
we could put them in one **module**, and give the module any name we want

To create a **module** in Python, open up an IDE or text editor,
create a file, and give it a descriptive name and a .py extension.

For this example, let's call it **sample.py** and enter in the following code:

<https://www.sitepoint.com/python-modules-packages/>

Creating a simple module

```
sample_variable = "This is a string variable in the sample.py module"
```

```
# A function in the module
```

```
def say_hello(name):
```

```
    return f"Hello, {name} welcome to this simple module."
```

```
# This is another function in the module
```

```
def add(a, b):
```

```
    return f"The sum of {a} + {b} is = {a+b}"
```

```
print(sample_variable)
```

```
print(say_hello("aaa"))
```

```
print(add(2, 3))
```

<https://www.sitepoint.com/python-modules-packages/>

Creating a simple module

The code above defines a module named `sample.py`.

It contains a variable named `sample_variable` whose value is the string
"This is a string variable in the `sample.py` module".

This module also contains two function definitions.

When called, the `say_hello()` function takes in a name parameter, and it returns a welcome message if we pass a name to it.

The `add()` function returns the sum of two numbers that have been passed to it.

<https://www.sitepoint.com/python-modules-packages/>

Creating a simple module

To run

```
python sample.py
```

```
python3 sample.py
```

This will return the following output:

This is a string variable in the sample.py module

Hello, aaa welcome to this simple module.

The sum of $2 + 3$ is = 5

For one-off module usage, we can run it as a standalone, but most modules are made to be used in other modules or other parts of a Python program.

So to use variables, functions, and classes from one module in another module we have to import the module.

<https://www.sitepoint.com/python-modules-packages/>

Using the import statement

We can use the import statement to make the contents of one module available for use in another module.

Consider our sample.py from above:

to use its contents in another module, we just import it:

```
# another_module.py

import sample

print(sample.sample_variable)
print(sample.say_hello("John"))
print(sample.add(2, 3))
```

The code above shows how to import the functions from the sample.py module, making them available for use in the another_module.py.

Note that, when we import a module, we don't include the .py extension:

<https://www.stepic.com/en/python-modules-packages/>

Python automatically knows we're importing a module.

Using the from statement

We can also use the from keyword to import specific functions or variables. Say a module has a large number of functions and variables defined in it and we don't want to use all of them. We can specify the functions or variables we want to use, using the from keyword:

```
# another_module.py  
  
from sample import add  
  
print(add(10, 4))
```

The code above shows that we've specifically imported the add() function from the sample module.

Another benefit of using the from keyword is that we'll run the imported function without namespacing it or prefixing it with the name of its parent module. Instead, we'll use the function like we've defined it in the file where we're using it. This leads to more concise and readable code.

<https://www.sitepoint.com/python-modules-packages/>

Using the as statement (1)

We can use `as` to provide an alias or an alternate name for the module.

At times, we may define module names that are quite long or unreadable. Python provides a way of giving the module imports an alternate or alias, which we can use to refer to them in the modules we're importing them into. To do this, we'll use the `as` keyword:

```
# another_module.py

import sample as sp

result = sp.add(5, 5)
print(result)
print(sp.say_hello("Jason"))
```

This code shows an import of the `sample` module, where the module is being given an alternate name `sp`. So using `sp` is just the same as calling `sample`. Therefore, using the alias, we have access to the variables and functions, in the same way we could if we were using the original name.

<https://www.sitepoint.com/python-modules-packages/>

Using the as statement (2)

Using those three methods, we're able to use the variables or functions from one module in another module, enhancing the readability of our application where we don't need to put the code in one file.

While naming our modules, it's good practice to use lowercase letters and separate words with underscores. For instance, if we have a module for handling database connections, we might name it `database_connection.py`. To avoid naming conflicts, try to choose descriptive and unique names for modules. If a module name might cause a name clash with a Python built-in keyword or module from a third-party library, consider using a different name or adding a prefix that's relevant to the project. Also, remember that names are case-sensitive in Python, so make sure to use the correct module name when importing.

Overall, using modules lets us create and organize our code in a readable and maintainable way. And this is very useful — whether we're working on a small script or a large application. Later, we'll look at some common Python standard library modules.

<https://www.sitepoint.com/python-modules-packages/>

Package

A package in Python is a way of organizing related modules into a directory. This provides a better way of organizing code, enabling us to group modules that serve a common purpose or are part of the same component.

Packages are particularly beneficial when structuring larger projects or libraries. For instance, consider the case of a web application where we have code for different database models, views, and utilities.

It would make a lot of sense if we created a models package with different modules for the different models in an application. Say our web app is a blogging application: possible models could be a users model and a posts model; we would then create a module for user management, and a module for posts management, and then put them in the models package.

It's important to reiterate at this point that modules are individual files containing Python code: they help put related functions, classes, and variables within a single file. In contrast, packages are directories that contain multiple modules or subpackages. They provide a higher level of organization for our code, by grouping related modules and enabling us to create more structured and maintainable projects.

<https://www.sitepoint.com/python-modules-packages/>

Building and managing packages

While packages organize related code modules in one directory, just putting the modules in a directory doesn't make it a package. For Python to identify a directory as a package or a subpackage, the directory must contain a special file named `__init__.py`.

This file notifies Python that the directory containing it should be treated as a package or a subpackage. This file could be empty, and most of the time it is, but it can also contain initialization code, and it plays a vital role in Python's package structure and import mechanisms. So using `__init__.py` tells Python that we are intentionally creating a package, thereby helping it differentiate between a package and an ordinary directory.

Packages can have a hierarchical structure, meaning we can create subpackages within our packages to further organize our code. This enables finer and more controlled separation of components and functionality. Consider the following example:

```
my_package/  
├── __init__.py  
├── module1.py  
└── subpackage/  
    ├── __init__.py  
    ├── submodule1.py  
    └── submodule2.py
```

<https://www.datacamp.com/python-modules/packages/>
This diagram shows `my_package` is the main package, and `subpackage` is a subpackage within it. Both directories have an `__init__.py` file. Using this kind of structure helps us organize our code into a meaningful hierarchy.

Creating packages and sub-packages (1)

To create a package, we first create a directory that's going to contain our modules. Then we create an `__init__.py` file. Then we create our modules in it, along with any subpackages.

Say we're building a calculator application: let's create a package for various calculations, so create a directory in our terminal or our IDE and name it calculator.

In the directory, create the `__init__.py` file, then create some modules. Let's create three modules, `add.py`, `subtract.py`, and `multiply.py`. In the end, we'll have a directory structure similar to this:

```
calculator/  
├── __init__.py  
├── add.py  
├── subtract.py  
└── multiply.py
```

<https://www.sitepoint.com/python-modules-packages/>

Creating packages and sub-packages (2)

Let's put some samples in those files. Open the add.py module and put in the following code:

```
# add.py

def add(a, b):
    """
    Adds two numbers and returns the result.

    :param a: First number.
    :param b: Second number.
    :return: Sum of a and b.
    """
    return a + b
```

This creates a module for addition, separating it from other calculations. Let's create one more module for subtraction. Open the subtract.py file and put the following code in it:

```
# subtract.py

def subtract(a, b):
    """
    Subtracts two numbers and returns the result.
```

<https://www.sitepoint.com/python-modules-packages/>

```
:param a: First number.
:param b: Second number.
return: Difference of a and b.
"""
return a - b
```

Importing from packages - absolute import

Absolute imports are used to directly import modules or subpackages from the top-level package, where we specify the full path to the module or package we want to import.

Here's an example of importing the add module from the calculator package:

```
# calculate.py  
  
from calculator.add import add  
  
result = add(5, 9)  
  
print(result)
```

The above example shows an external module — calculate.py — that imports the add() function from the add module using an absolute import by specifying the absolute path to the function.

<https://www.sitepoint.com/python-modules-packages/>

Importing from packages - relative import

Relative imports are used to import modules or packages relative to the current module's position in the package hierarchy. Relative imports are specified using dots (.) to indicate the level of relative positioning.

In order to demonstrate relative imports, let's create a subpackage in the calculator package, call the subpackage multiply, then move the multiply.py module into that subpackage, so that we'll have an updated package structure like this:

```
calculator/  
├── __init__.py  
├── add.py  
├── subtract.py  
└── multiply/  
    ├── __init__.py  
    └── multiply.py
```

With this setup, we can now use relative imports to access the multiply module from other modules within the calculator package or its subpackages. For instance, if we had a module inside the calculator package that needs to import the multiply module, we could use the code below:

```
from .multiply import multiply
```

```
result = multiply(5, 9)  
print(result)
```

<https://www.sitepoint.com/python-modules-packages/>

Package (1)

modules are
files containing Python statements and definitions,
like function and class definitions.

to bundle multiple **modules** together,
create a **package**.

a **package** is
basically a **directory**
with several **Python files** (**modules**)
and a special file **`__init__.py`**

inside of the **Python path**,
every **directory** contains **`__init__.py`**,
will be treated as a **package** by Python.

<https://python-course.eu/python-tutorial/packages.php>

Submodules in a package

packages are a way of structuring Python's **module namespace** by using "**dotted module names**".

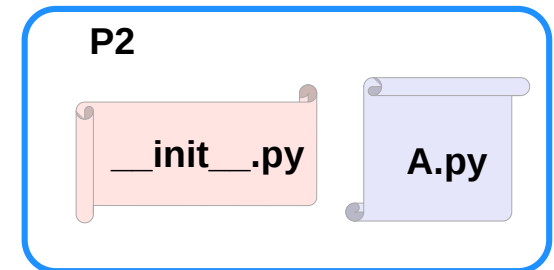
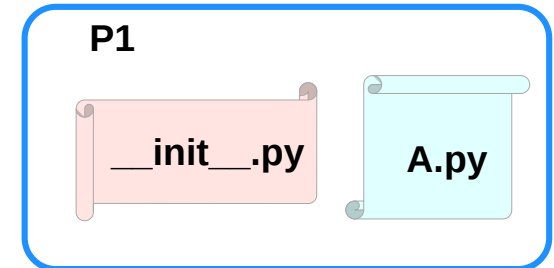
A.B stands for a **submodule** named **B** in a **package** named **A**.

two different **packages** like **P1** and **P2** can both have **modules** with the same name, let's say **A**, for example.

The **submodule A** of the **package P1** and the **submodule A** of the **package P2** can be totally different.

P1.A
P2.A

A **package** is imported like a "normal" **module**.



<https://python-course.eu/python-tutorial/packages.php>

Creating a package

to create a **package**, we need a **directory**.

the **name** of this **directory** will be the **name** of the **package**,

assume we want to create "**simple_package**" package

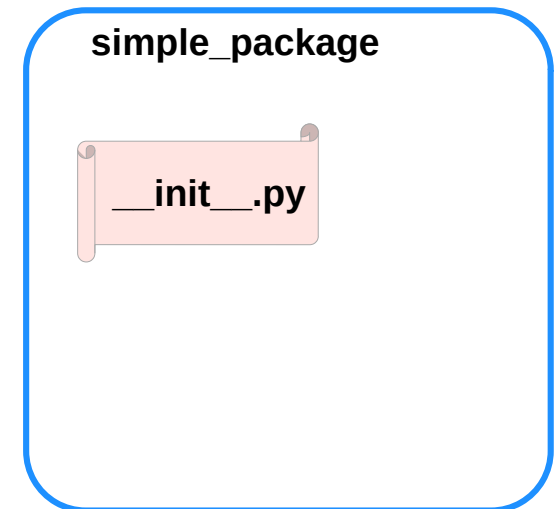
must create directory "**simple_package**" and this directory needs to contain the "**`__init__.py`**" file

this file can be **empty**, or can contain valid **Python code**.

this **code** will be **executed** when a **package** is **imported**,

so it can be used to **initialize** a **package**,

e.g. to make sure that some other modules are **imported** or some values **set**.



<https://python-course.eu/python-tutorial/packages.php>

Examples of creating a package (1)

put all of the **Python files** which will be the **submodules** into the **directory** for a **package**.

create two simple files **a.py** and **b.py**

a.py: → **submodule a**

```
def bar():  
    print("Hello, function 'bar' from module 'a' calling")
```

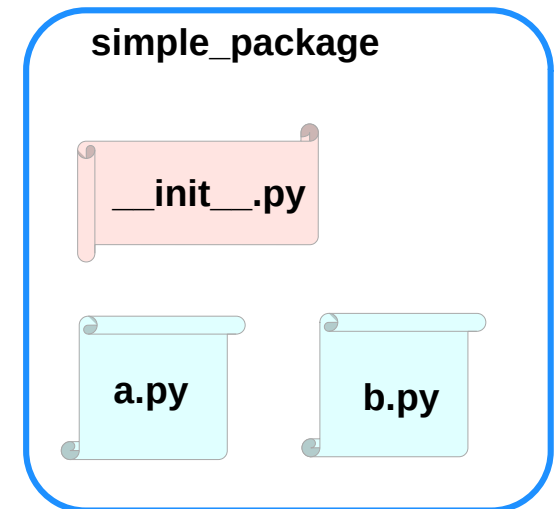
b.py: → **submodule b**

```
def foo():  
    print("Hello, function 'foo' from module 'b' calling")
```

an empty file with the name **__init__.py** inside of **simple_package** directory

__init__.py:

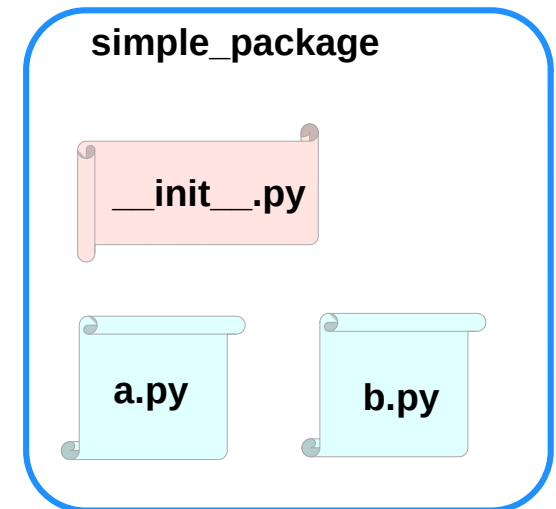
empty file



<https://python-course.eu/python-tutorial/packages.php>

Examples of creating a package (2)

`import simple_package` from the interactive Python shell,
assuming that the directory `simple_package` is
either in the directory from which you call the shell or
that it is contained in the `search path` or
environment variable "`PYTHONPATH`" (from your operating system):



<https://python-course.eu/python-tutorial/packages.php>

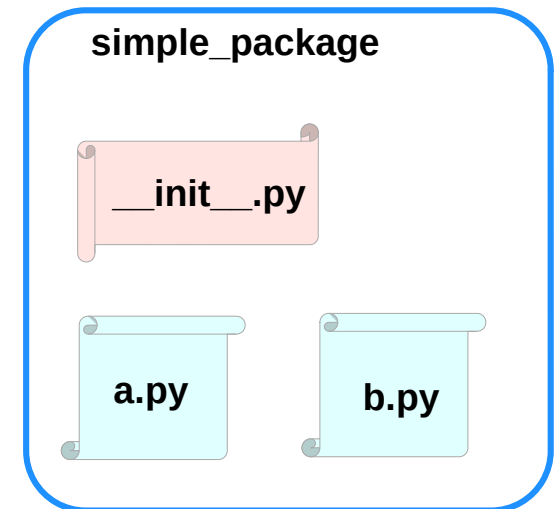
Examples of creating a package (3)

```
import simple_package
simple_package/a
```

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-3-347df8a711cc> in <module>
----> 1 simple_package/a
NameError: name 'a' is not defined
```

```
simple_package/b
```

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-4-e71d2904d2bd> in <module>
----> 1 simple_package/b
NameError: name 'b' is not defined
```



<https://python-course.eu/python-tutorial/packages.php>

Examples of creating a package (4)

the **package** `simple_package` has been loaded
but neither the **module** `"a"` nor the **module** `"b"` has been loaded

can't access neither `"a"` nor `"b"`
by solely importing `simple_package`.

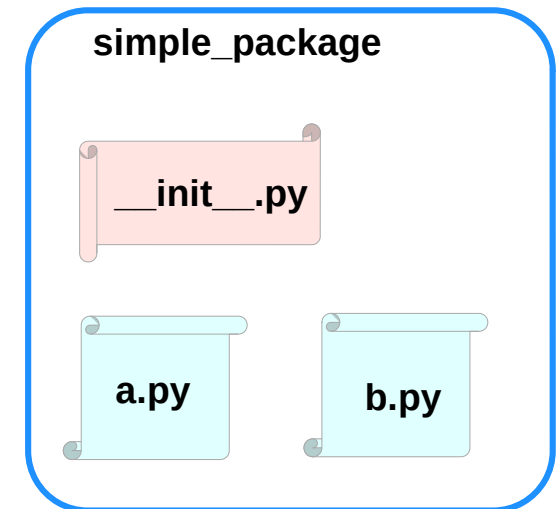
must import the **modules** `a` and `b` as follows

```
from simple_package import a, b
```

```
a.bar()
```

```
b.foo()
```

Hello, function 'bar' from module 'a' calling
Hello, function 'foo' from module 'b' calling



<https://python-course.eu/python-tutorial/packages.php>

Examples of creating a package (5)

to automatically load these **modules**.

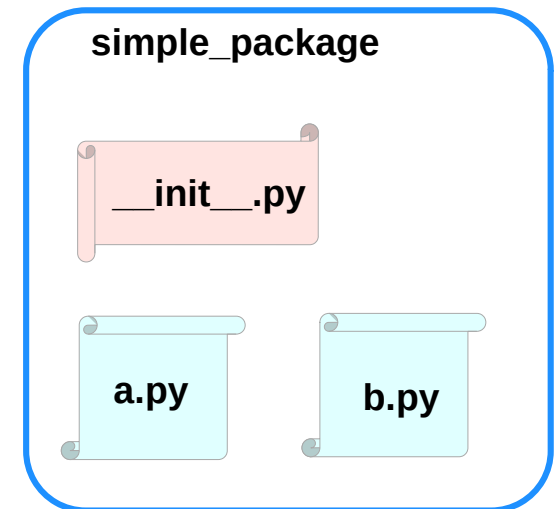
add the following lines to the file `__init__.py`:

```
import simple_package.a
import simple_package.b
```

Then

```
import simple_package
simple_package.a.bar()
simple_package.b.foo()
```

Hello, function 'bar' from module 'a' calling
Hello, function 'foo' from module 'b' calling



<https://python-course.eu/python-tutorial/packages.php>

Package Examples (1)

```
sound
|-- effects
|   |-- __init__.py
|   |-- echo.py
|   |-- reverse.py
|   |-- surround.py
|-- filters
|   |-- __init__.py
|   |-- equalizer.py
|   |-- karaoke.py
|   |-- vocoder.py
|-- formats
|   |-- __init__.py
|   |-- aiffread.py
|   |-- aiffwrite.py
|   |-- auread.py
|   |-- auwrite.py
|   |-- wavread.py
|   |-- wavwrite.py
|-- __init__.py
```

sound

__init__.py

effects

__init__.py

echo.py
reverse.py
surround.py

filters

__init__.py

equalizer.py
karaoke.py
vocoder.py

formats

__init__.py

aiffread.py aurwrite.py
aiffwrite.py wavred.py
auread.py wavwrite.py

<https://python-course.eu/python-tutorial/packages.php>

Package Examples sound1

sound1

`__init__.py`

effects

`__init__.py`

`echo.py`
`reverse.py`
`surround.py`

formats

`__init__.py`

`aiffread.py` `aurwrite.py`
`aifwrite.py` `wavred.py`
`auread.py` `wavwrite.py`

filters

`__init__.py`

`equalizer.py`
`karaoke.py`
`vocoder.py`

```
import sound1
print(sound1)                    ... OK
print(sound1.effects)         ... Error
```

```
import sound1.effects
print(sound1.effects)         ... OK
```

<https://python-course.eu/python-tutorial/packages.php>

Package Examples sound2

sound2

`__init__.py`

`import sound2.effects`

effects

`__init__.py`

`echo.py`
`reverse.py`
`surround.py`

formats

`__init__.py`

`aiffread.py` `aurwrite.py`
`aifwrite.py` `wavred.py`
`auread.py` `wavwrite.py`

filters

`__init__.py`

`equalizer.py`
`karaoke.py`
`vocoder.py`

```
import sound2
print(sound2) ... OK
print(sound2.effects) ... OK
```

<https://python-course.eu/python-tutorial/packages.php>

Package Examples sound3

sound3

`__init__.py`

`from . import effects`

effects

`__init__.py`

`echo.py`
`reverse.py`
`surround.py`

formats

`__init__.py`

`aiffread.py` `aurwrite.py`
`aifwrite.py` `wavred.py`
`auread.py` `wavwrite.py`

filters

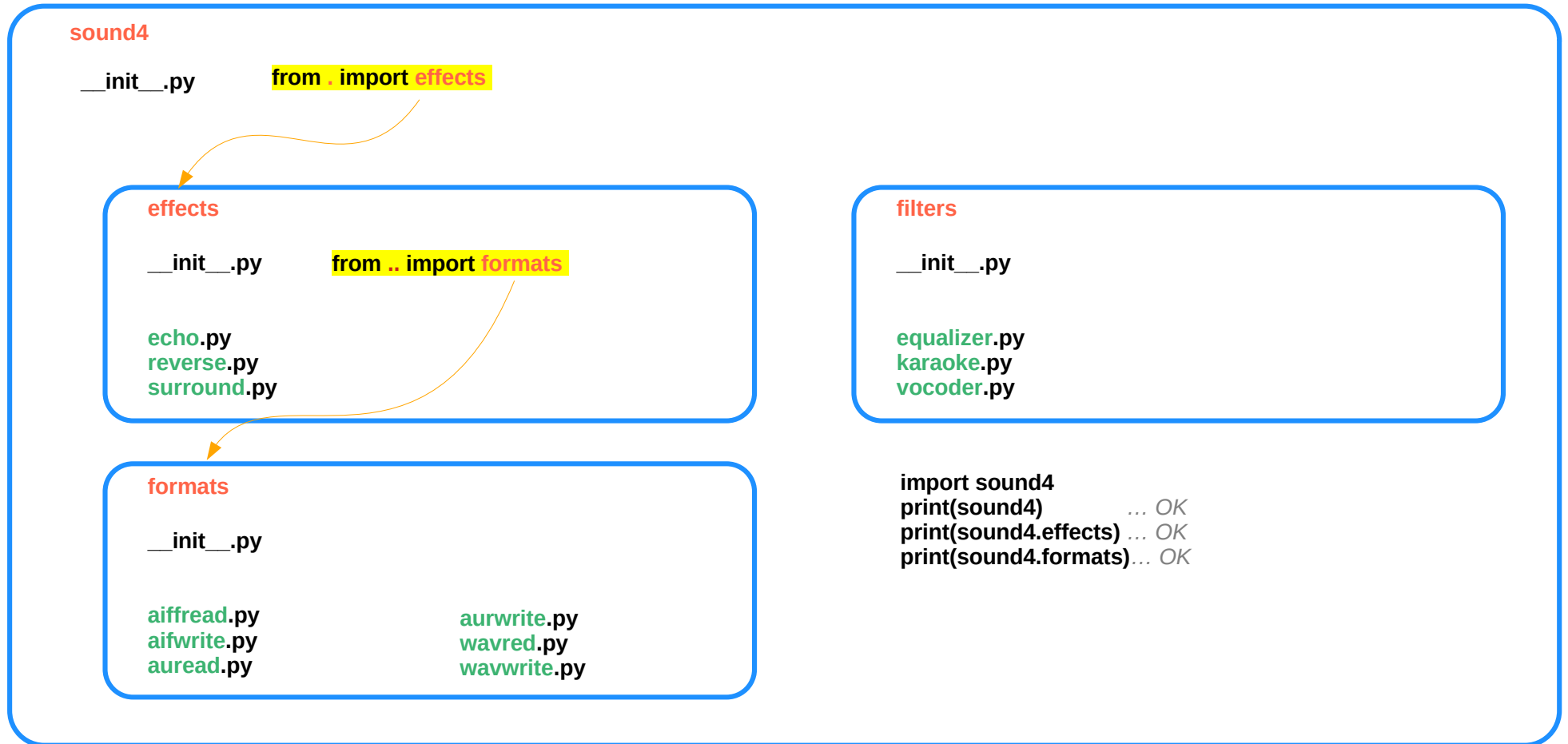
`__init__.py`

`equalizer.py`
`karaoke.py`
`vocoder.py`

```
import sound3
print(sound3)                      ... OK
print(sound3.effects)              ... OK
```

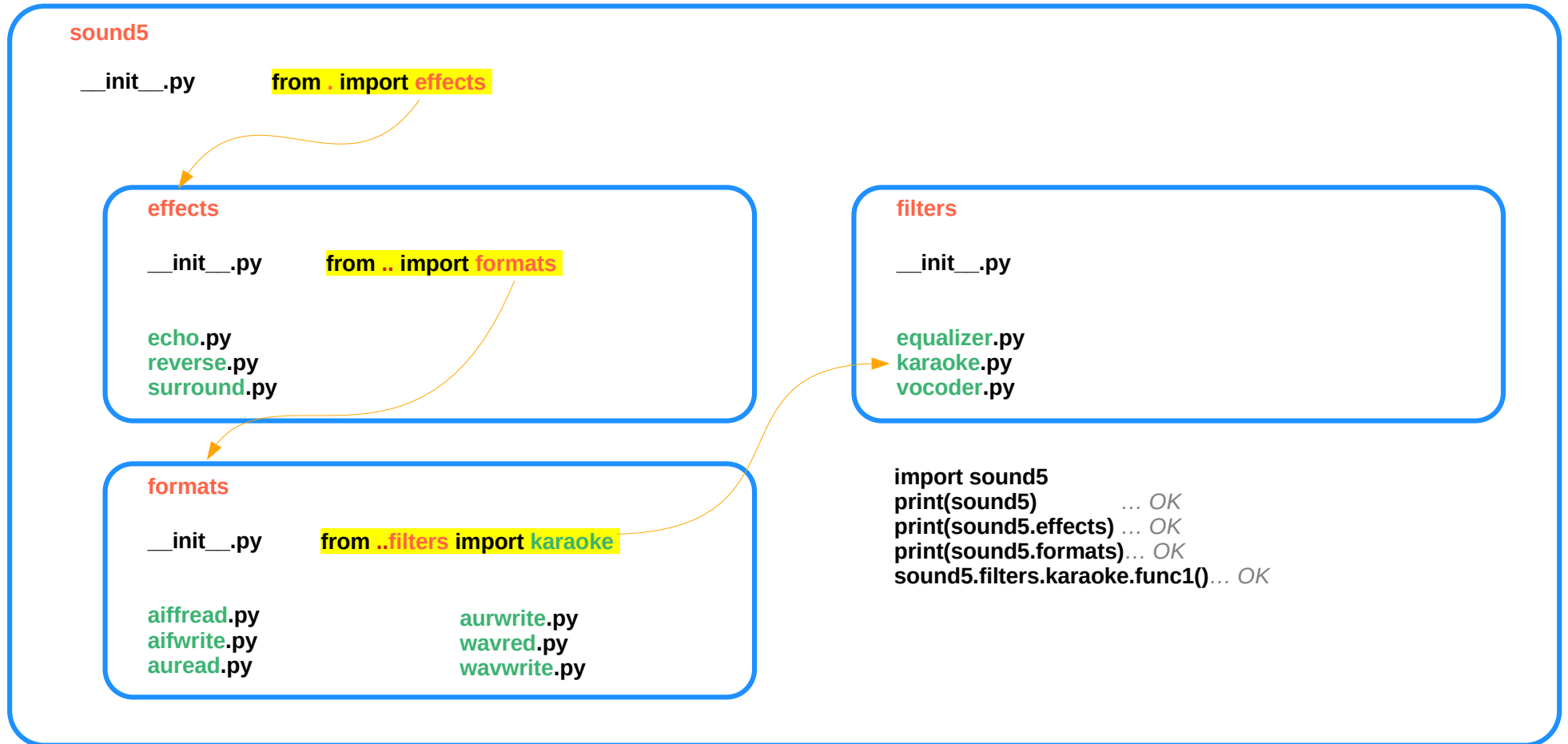
<https://python-course.eu/python-tutorial/packages.php>

Package Examples sound4



<https://python-course.eu/python-tutorial/packages.php>

Package Examples sound5



<https://python-course.eu/python-tutorial/packages.php>

Package Examples sound6

sound6

`__init__.py`

`foobar.py` *empty file*

effects

`__init__.py`

`echo.py`
`reverse.py`
`surround.py`

formats

`__init__.py`

`aiffread.py` `aurwrite.py`
`aifwrite.py` `wavred.py`
`auread.py` `wavwrite.py`

filters

`__init__.py`

`equalizer.py`
`karaoke.py`
`vocoder.py`

```
from sound6 import *
```

sound6 package is getting imported!

```
for mod in ['foobar', 'effects', 'filters', 'formats']:  
    print(mod, mod in dir())  
foobar False  
effects False  
filters False  
formats False
```

<https://python-course.eu/python-tutorial/packages.php>

Package Examples sound7

sound7

```
__init__.py __all__ = ["effects", "filters", "formats", "foobar"]
```

```
foobar.py empty file
```

effects

```
__init__.py
```

```
echo.py  
reverse.py  
surround.py
```

*sound6 package
effects package
filters package
formats package
foobar module*

filters

```
__init__.py
```

```
equalizer.py  
karaoke.py  
vocoder.py
```

formats

```
__init__.py
```

```
aiffread.py  
aifwrite.py  
auread.py  
aurwrite.py  
wavred.py  
wavwrite.py
```

```
from sound7 import *
```

sound7 package is getting imported!

```
for mod in ['foobar', 'effects', 'filters', 'formats']:  
    print(mod, mod in dir())  
    foobar True  
    effects True  
    filters True  
    formats True
```

<https://python-course.eu/python-tutorial/packages.php>

Package Examples sound8

sound7

```
__init__.py all_ = ["effects", "filters", "formats", "foobar"]  
foobar.py empty file
```

effects

```
__init__.py  
all_ = ["echo", "reverse", "surround"]
```

```
echo.py  
reverse.py  
surround.py
```

filters

```
__init__.py  
all_ = ["equalizer", "karaoke", "vocoder", "__init__"]
```

```
equalizer.py  
karaoke.py  
vocoder.py
```

formats

```
__init__.py  
all_ = ["aiffread", "aifwrite", "auread",  
        "aurwrite", "wavred", "wavwrite"]
```

```
aiffread.py      aurwrite.py  
aifwrite.py     wavred.py  
auread.py       wavwrite.py
```

```
from sound8 import *  
sound8 package is getting imported!
```

```
from sound8.effects import *  
xxx package is getting imported!
```

```
from sound8.filters import *  
xxx package is getting imported!
```

```
from sound8.formats import *  
xxx package is getting imported!
```

<https://python-course.eu/python-tutorial/packages.php>

Package sound1 (1)

```
__init__.py
print("sound1 package is getting imported!")
```

```
effects/__init__.py
print("effects package is getting imported!")
```

```
effects/echo.py
def func1():
    print("Function func1 has been called!")
    print("Module echo.py has been loaded!")
```

```
effects/reverse.py
def func1():
    print("Function func1 has been called!")
    print("Module reverse.py has been loaded!")
```

```
effects/surround.py
def func1():
    print("Function func1 has been called!")
```

```
sound
|-- effects
|-- filters
|-- formats
|-- __init__.py
```

```
filters/__init__.py
print("filters package is getting imported!")
```

```
filters/equalizer.py
def func1():
    print("Function func1 has been called!")
    print("Module equalizer.py has been loaded!")
```

```
filters/karaoke.py
def func1():
    print("Function func1 has been called!")
    print("Module karaoke.py has been loaded!")
```

```
filters/vocoder.py
def func1():
    print("Function func1 has been called!")
    print("Module vocoder.py has been loaded!")
```

If we import the package `sound1` by using the statement `import sound1`, Only the package `sound1` is imported but none of the subpackages will be imported `effects`, `filters` and `formats`

because the file `__init__.py` doesn't contain any code for importing subpackages:

```
import sound1
print(sound1)           ... OK
print(sound1.effects)  ... Error
```

```
formats/__init__.py
print("formats package is getting imported!")
```

```
formats/aiffread.py
def func1():
    print("Function func1 has been called!")
    print("Module aiffread.py has been loaded!")
```

```
formats/aiffwrite.py
def func1():
    print("Function func1 has been called!")
    print("Module aiffwrite.py has been loaded!")
```

```
formats/auread.py
def func1():
    print("Function func1 has been called!")
    print("Module auread.py has been loaded!")
```

```
formats/auwrite.py
def func1():
    print("Function func1 has been called!")
    print("Module auwrite.py has been loaded!")
```

```
formats/wavread.py
def func1():
    print("Function func1 has been called!")
    print("Module wavread.py has been loaded!")
```

```
formats/wavwrite.py
def func1():
    print("Function func1 has been called!")
    print("Module wavwrite.py has been loaded!")
```

<https://python-course.eu/python-tutorial/packages.php>

Package sound1 (2)

```
import sound1
print(sound1)
print(sound1.effects)
```

OUTPUT:

```
<module 'sound1' from '/data/Dropbox (Bodenseo)/
Bodenseo Team Folder/melisa/notebooks_en/
sound1/__init__.py'>
```

```
AttributeError                                Traceback (most recent call last)
<ipython-input-2-0b6d7fed3b24> in <module>
      3 print(sound1)
      4
----> 5 print(sound1.effects)
AttributeError: module 'sound1' has no attribute 'effects'
```

print(sound1)

print(sound1.effects)

If you also want to use the package **effects**, you have to import it explicitly with `import sound.effects`:

```
import sound1.effects
print(sound1.effects)
```

```
<module 'sound1.effects' from '/data/Dropbox (Bodenseo)/
Bodenseo Team Folder/melisa/notebooks_en/
sound1/effects/__init__.py'>
```

It is possible to have the submodule importing done automatically when importing the sound1 module.

We will change now to **sound2** to demonstrate how to do this.

We use the same files as in **sound1**, but we will add the code line `import sound2.effects` into the file `__init__.py` of the directory **sound2**.

```
"""An empty sound package
This is the sound package, providing hardly anything!"""
import sound2.effects
print("sound2.effects package is getting imported!")
)
```

<https://python-course.eu/python-tutorial/packages.php>

Package sound2

```
__init__.py  
print("sound2 package is getting imported!")  
import sound2.effects
```

```
effects/__init__.py  
print("effects package is getting imported!")
```

```
effects/echo.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module echo.py has been loaded!")
```

```
effects/reverse.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module reverse.py has been loaded!")
```

```
effects/surround.py  
def func1():  
    print("Function func1 has been called!")
```

```
sound  
|-- effects  
|-- filters  
|-- formats  
|-- __init__.py
```

```
filters/__init__.py  
print("filters package is getting imported!")
```

```
filters/equalizer.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module equalizer.py has been loaded!")
```

```
filters/karaoke.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module karaoke.py has been loaded!")
```

```
filters/vocoder.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module vocoder.py has been loaded!")
```

```
import sound2.effects  
in __init__.py of the package sound2
```

when the package `sound2` is imported,
the subpackage `effects` will also
be automatically loaded:

```
import sound2  
sound2 package is getting imported!  
effects package is getting imported!
```

```
formats/__init__.py  
print("formats package is getting imported!")
```

```
formats/aiffread.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module aiffread.py has been loaded!")
```

```
formats/aiffwrite.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module aiffwrite.py has been loaded!")
```

```
formats/auread.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module auread.py has been loaded!")
```

```
formats/auwrite.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module auwrite.py has been loaded!")
```

```
formats/wavread.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module wavread.py has been loaded!")
```

```
formats/wavwrite.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module wavwrite.py has been loaded!")
```

<https://python-course.eu/python-tutorial/packages.php>

Package sound3

```
__init__.py  
print("sound3 package is getting imported!")  
from . import effects
```

```
effects/__init__.py  
print("effects package is getting imported!")
```

```
effects/echo.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module echo.py has been loaded!")
```

```
effects/reverse.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module reverse.py has been loaded!")
```

```
effects/surround.py  
def func1():  
    print("Function func1 has been called!")
```

```
sound  
|-- effects  
|-- filters  
|-- formats  
|-- __init__.py
```

```
filters/__init__.py  
print("filters package is getting imported!")
```

```
filters/equalizer.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module equalizer.py has been loaded!")
```

```
filters/karaoke.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module karaoke.py has been loaded!")
```

```
filters/vocoder.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module vocoder.py has been loaded!")
```

Instead of using an **absolute path** we could have imported the **effects** package **relative** to the **sound2** package.

```
import sound2.effects # absolute path
```

```
from . import effects # relative path
```

```
import sound3  
sound3 package is getting imported!  
effects package is getting imported!
```

```
formats/__init__.py  
print("formats package is getting imported!")
```

```
formats/aiffread.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module aiffread.py has been loaded!")
```

```
formats/aiffwrite.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module aiffwrite.py has been loaded!")
```

```
formats/auread.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module auread.py has been loaded!")
```

```
formats/auwrite.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module auwrite.py has been loaded!")
```

```
formats/wavread.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module wavread.py has been loaded!")
```

```
formats/wavwrite.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module wavwrite.py has been loaded!")
```

<https://python-course.eu/python-tutorial/packages.php>

Package sound4

```
__init__.py  
print("sound4 package is getting imported!")  
from . import effects
```

```
effects/__init__.py  
print("effects package is getting imported!")  
from .. import formats
```

```
effects/__init__.py  
print("effects package is getting imported!")
```

```
effects/echo.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module echo.py has been loaded!")
```

```
effects/reverse.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module reverse.py has been loaded!")
```

```
effects/surround.py  
def func1():  
    print("Function func1 has been called!")
```

```
sound  
|-- effects  
|-- filters  
|-- formats  
|-- __init__.py
```

```
filters/__init__.py  
print("filters package is getting imported!")
```

```
filters/equalizer.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module equalizer.py has been loaded!")
```

```
filters/karaoke.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module karaoke.py has been loaded!")
```

```
filters/vocoder.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module vocoder.py has been loaded!")
```

```
import sound4  
sound4 package is getting imported!  
effects package is getting imported!  
formats package is getting imported!
```

in the `__init__.py` file of **sound4** directory

```
from . import effects
```

in the `__init__.py` file of **effects** directory

```
from .. import formats
```

```
formats/__init__.py  
print("formats package is getting imported!")
```

```
formats/aiffread.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module aiffread.py has been loaded!")
```

```
formats/aiffwrite.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module aiffwrite.py has been loaded!")
```

```
formats/auread.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module auread.py has been loaded!")
```

```
formats/auwrite.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module auwrite.py has been loaded!")
```

```
formats/wavread.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module wavread.py has been loaded!")
```

```
formats/wavwrite.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module wavwrite.py has been loaded!")
```

<https://python-course.eu/python-tutorial/packages.php>

Package sound5

```
__init__.py
print("sound5 package is getting imported!")
from . import effects
```

```
effects/__init__.py
print("effects package is getting imported!")
from .. import formats
```

```
effects/echo.py
def func1():
    print("Function func1 has been called!")
    print("Module echo.py has been loaded!")
```

```
effects/reverse.py
def func1():
    print("Function func1 has been called!")
    print("Module reverse.py has been loaded!")
```

```
effects/surround.py
def func1():
    print("Function func1 has been called!")
```

```
sound
|-- effects
|-- filters
|-- formats
|-- __init__.py
```

```
filters/__init__.py
print("filters package is getting imported!")
```

```
filters/equalizer.py
def func1():
    print("Function func1 has been called!")
    print("Module equalizer.py has been loaded!")
```

```
filters/karaoke.py
def func1():
    print("Function func1 has been called!")
    print("Module karaoke.py has been loaded!")
```

```
filters/vocoder.py
def func1():
    print("Function func1 has been called!")
    print("Module vocoder.py has been loaded!")
```

import karaoke module
from filters package
when we import the effects package.

from ..filters import karaoke
into the `__init__.py` file of formats directory

can access the functions of karaoke :

```
sound5.filters.karaoke.func1()
```

```
Function func1 has been called!
```

```
formats/__init__.py
print("formats package is getting imported!")
from ..filters import karaoke
```

```
formats/aiffread.py
def func1():
    print("Function func1 has been called!")
    print("Module aiffread.py has been loaded!")
```

```
formats/aiffwrite.py
def func1():
    print("Function func1 has been called!")
    print("Module aiffwrite.py has been loaded!")
```

```
formats/auread.py
def func1():
    print("Function func1 has been called!")
    print("Module auread.py has been loaded!")
```

```
formats/auwrite.py
def func1():
    print("Function func1 has been called!")
    print("Module auwrite.py has been loaded!")
```

```
formats/wavread.py
def func1():
    print("Function func1 has been called!")
    print("Module wavread.py has been loaded!")
```

```
formats/wavwrite.py
def func1():
    print("Function func1 has been called!")
    print("Module wavwrite.py has been loaded!")
```

<https://python-course.eu/python-tutorial/packages.php>

Package sound6 (1)

```
__init__.py  
print("sound5 package is getting imported!")
```

```
foobar.py  
empty file
```

```
effects/__init__.py  
print("effects package is getting imported!")
```

```
effects/echo.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module echo.py has been loaded!")
```

```
effects/reverse.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module reverse.py has been loaded!")
```

```
effects/surround.py  
def func1():  
    print("Function func1 has been called!")
```

```
sound  
|-- effects  
|-- filters  
|-- formats  
|-- __init__.py  
|-- foobar.py
```

```
filters/__init__.py  
print("filters package is getting imported!")
```

```
filters/equalizer.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module equalizer.py has been loaded!")
```

```
filters/karaoke.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module karaoke.py has been loaded!")
```

```
filters/vocoder.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module vocoder.py has been loaded!")
```

```
from sound6 import *  
    sound6 package is getting imported!  
  
for mod in  
    ['foobar', 'effects', 'filters', 'formats']:  
    print(mod, mod in dir())  
    foobar False  
    effects False  
    filters False  
    formats False
```

```
formats/__init__.py  
print("formats package is getting imported!")
```

```
formats/aiffread.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module aiffread.py has been loaded!")
```

```
formats/aiffwrite.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module aiffwrite.py has been loaded!")
```

```
formats/auread.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module auread.py has been loaded!")
```

```
formats/auwrite.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module auwrite.py has been loaded!")
```

```
formats/wavread.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module wavread.py has been loaded!")
```

```
formats/wavwrite.py  
def func1():  
    print("Function func1 has been called!")  
    print("Module wavwrite.py has been loaded!")
```

<https://python-course.eu/python-tutorial/packages.php>

Package sound6 (2)

add a module (file) **foobar** (filename: **foobar.py**) to the **sound** directory.

want to import all the **submodules** and **subpackages** of the package **sound6**.

```
from sound6 import *
```

sound6 package is getting imported!

Yet, if we check with the **dir** function, we see that neither the **module foobar** nor the **subpackages effects, filters** and **formats** have been imported:

```
for mod in ['foobar', 'effects', 'filters', 'formats']:  
    print(mod, mod in dir())
```

```
foobar False  
effects False  
filters False  
formats False
```

```
sound7  
|-- effects  
|-- filters  
|-- formats  
|-- __init__.py  
|-- foobar.py
```

<https://python-course.eu/python-tutorial/packages.php>

Package sound7 (1)

```
__init__.py
print("sound5 package is getting imported!")
__all__ = ["formats", "filters", "effects",
           "foobar"]
```

```
foobar.py
empty file
```

```
effects/__init__.py
print("effects package is getting imported!")
```

```
effects/echo.py
def func1():
    print("Function func1 has been called!")
    print("Module echo.py has been loaded!")
```

```
effects/reverse.py
def func1():
    print("Function func1 has been called!")
    print("Module reverse.py has been loaded!")
```

```
effects/surround.py
def func1():
    print("Function func1 has been called!")
```

```
sound7
|-- effects
|-- filters
|-- formats
|-- __init__.py
|-- foobar.py
```

```
filters/__init__.py
print("filters package is getting imported!")
```

```
filters/equalizer.py
def func1():
    print("Function func1 has been called!")
    print("Module equalizer.py has been loaded!")
```

```
filters/karaoke.py
def func1():
    print("Function func1 has been called!")
    print("Module karaoke.py has been loaded!")
```

```
filters/vocoder.py
def func1():
    print("Function func1 has been called!")
    print("Module vocoder.py has been loaded!")
```

```
from sound7 import *           ... OK
```

```
from sound8.effects import *  ... not OK
```

```
formats/__init__.py
print("formats package is getting imported!")
```

```
formats/aiffread.py
def func1():
    print("Function func1 has been called!")
    print("Module aiffread.py has been loaded!")
```

```
formats/aiffwrite.py
def func1():
    print("Function func1 has been called!")
    print("Module aiffwrite.py has been loaded!")
```

```
formats/auread.py
def func1():
    print("Function func1 has been called!")
    print("Module auread.py has been loaded!")
```

```
formats/auwrite.py
def func1():
    print("Function func1 has been called!")
    print("Module auwrite.py has been loaded!")
```

```
formats/wavread.py
def func1():
    print("Function func1 has been called!")
    print("Module wavread.py has been loaded!")
```

```
formats/wavwrite.py
def func1():
    print("Function func1 has been called!")
    print("Module wavwrite.py has been loaded!")
```

<https://python-course.eu/python-tutorial/packages.php>

Package sound7 (2)

explicit index for the **subpackages** and **modules** of a **package**, which should be imported.

define a **list** named `__all__` to the `__init__.py` file of the **sound** directory.

: the **list** of **module** and **package** names to be imported when `from package import *` is encountered.

```
__all__ = ["formats", "filters", "effects", "foobar"]
```

```
from sound7 import *
```

```
sound7 package is getting imported!  
formats package is getting imported!  
filters package is getting imported!  
effects package is getting imported!  
foobar module is getting imported
```

```
sound7  
|-- effects  
|-- filters  
|-- formats  
|-- __init__.py  
|-- foobar.py
```

<https://python-course.eu/python-tutorial/packages.php>

check with `dir` again:

```
for mod in ['foobar', 'effects', 'filters', 'formats']:  
    print(mod, mod in dir())
```

```
foobar True  
effects True  
filters True  
formats True
```

if we use `*` in a **subpackage effects**

```
from sound.effects import *  
sound7 package is getting imported!  
effects package is getting imported!
```

```
dir()  
['__builtins__', '__doc__', '__loader__', '__name__',  
 '__package__', '__spec__']
```

Like expected the modules inside of **effects** have not been imported automatically.

Package sound8 (1)

```
__init__.py
print("sound8 package is getting imported!")
__all__ = ["formats", "filters", "effects",
           "foobar"]
```

`foobar.py`
empty file

```
effects/__init__.py
print("effects package is getting imported!")
__all__ = ["echo", "surround", "reverse"]
```

```
effects/echo.py
def func1():
    print("Function func1 has been called!")
    print("Module echo.py has been loaded!")
```

```
effects/reverse.py
def func1():
    print("Function func1 has been called!")
    print("Module reverse.py has been loaded!")
```

```
effects/surround.py
def func1():
    print("Function func1 has been called!")
```

```
sound8
|-- effects
|-- filters
|-- formats
|-- __init__.py
|-- foobar.py
```

```
filters/__init__.py
print("filters package is getting imported!")
__all__ = ["equalizer", "__init__", "karaoke",
           "vocoder"]
```

```
filters/equalizer.py
def func1():
    print("Function func1 has been called!")
    print("Module equalizer.py has been loaded!")
```

```
filters/karaoke.py
def func1():
    print("Function func1 has been called!")
    print("Module karaoke.py has been loaded!")
```

```
filters/vocoder.py
def func1():
    print("Function func1 has been called!")
    print("Module vocoder.py has been loaded!")
```

```
from sound8 import * ... OK
from sound8.effects import * ... OK
from sound8.filters import * ... OK
from sound8.formats import * ... OK
```

```
formats/__init__.py
print("formats package is getting imported!")
__all__ = ["aiffread", "aiffwrite", "auread",
           "auwrite", "wavread", "wavwrite"]
```

```
formats/aiffread.py
def func1():
    print("Function func1 has been called!")
    print("Module aiffread.py has been loaded!")
```

```
formats/aiffwrite.py
def func1():
    print("Function func1 has been called!")
    print("Module aiffwrite.py has been loaded!")
```

```
formats/auread.py
def func1():
    print("Function func1 has been called!")
    print("Module auread.py has been loaded!")
```

```
formats/auwrite.py
def func1():
    print("Function func1 has been called!")
    print("Module auwrite.py has been loaded!")
```

```
formats/wavread.py
def func1():
    print("Function func1 has been called!")
    print("Module wavread.py has been loaded!")
```

```
formats/wavwrite.py
def func1():
    print("Function func1 has been called!")
    print("Module wavwrite.py has been loaded!")
```

<https://python-course.eu/python-tutorial/packages.php>

Package sound8 (2)

`__all__` list in the `__init__` file of each sub-package

```
__all__ = ["equalizer", "__init__", "karaoke", "vocoder"]  
__all__ = ["aiffread", "aiffwrite", "auread", "auwrite",  
           "wavread", "wavwrite"]  
__all__ = ["echo", "surround", "reverse"]
```

```
from sound8 import *
```

sound8 package is getting imported!
formats package is getting imported!
filters package is getting imported!
effects package is getting imported!
foobar module is getting imported

```
from sound8.effects import *
```

Module `echo.py` has been loaded!
Module `surround.py` has been loaded!
Module `reverse.py` has been loaded!

```
from sound8.filters import *
```

Module `equalizer.py` has been loaded!
Module `karaoke.py` has been loaded!
Module `vocoder.py` has been loaded!

```
from sound8.formats import *
```

Module `aiffread.py` has been loaded!
Module `aiffwrite.py` has been loaded!
Module `auread.py` has been loaded!
Module `auwrite.py` has been loaded!
Module `wavread.py` has been loaded!
Module `wavwrite.py` has been loaded!

Although certain modules are designed to export only names that follow certain patterns when you use `import`, it is still considered bad practice.

The recommended way is to import specific modules from a package instead of using `*`

<https://python-course.eu/python-tutorial/packages.php>

Package sound6 (3)



<https://python-course.eu/python-tutorial/packages.php>