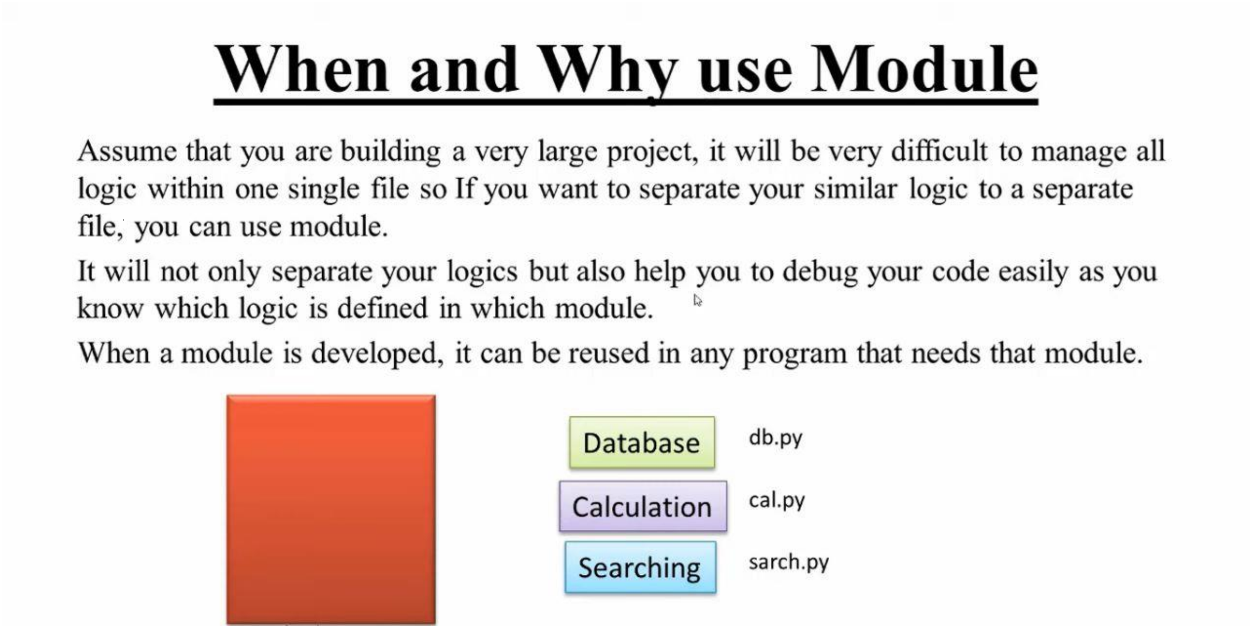


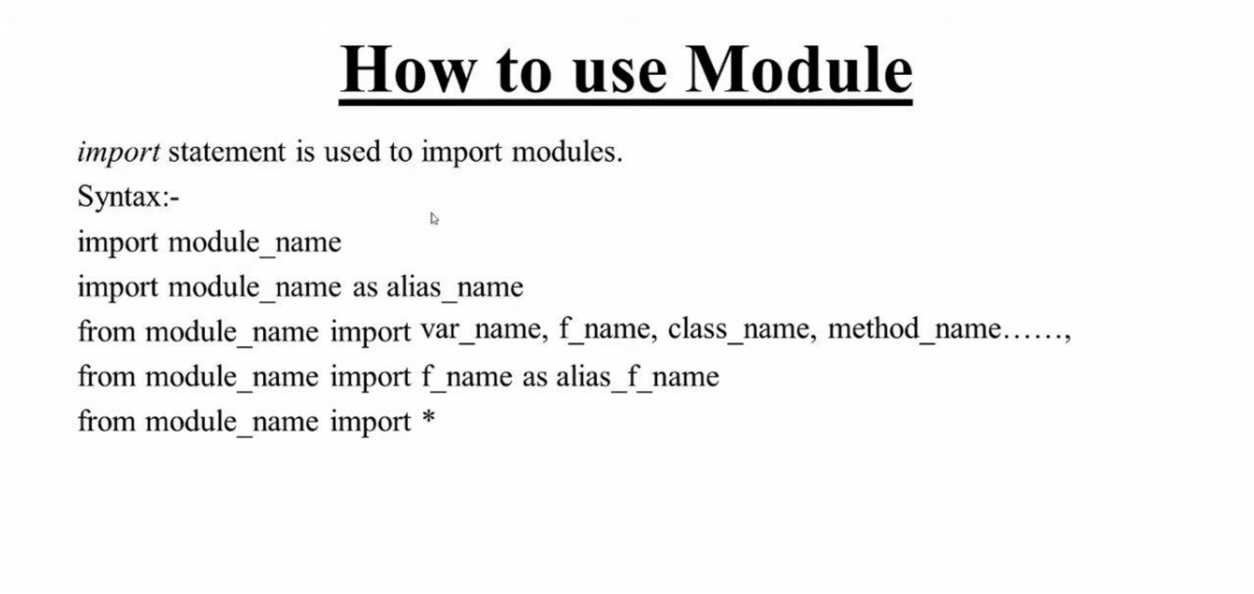
When and Why use Module

Assume that you are building a very large project, it will be very difficult to manage all logic within one single file so If you want to separate your similar logic to a separate file, you can use module.

It will not only separate your logics but also help you to debug your code easily as you know which logic is defined in which module.

When a module is developed, it can be reused in any program that needs that module.

main.py



How to use Module

import statement is used to import modules.

Syntax:-

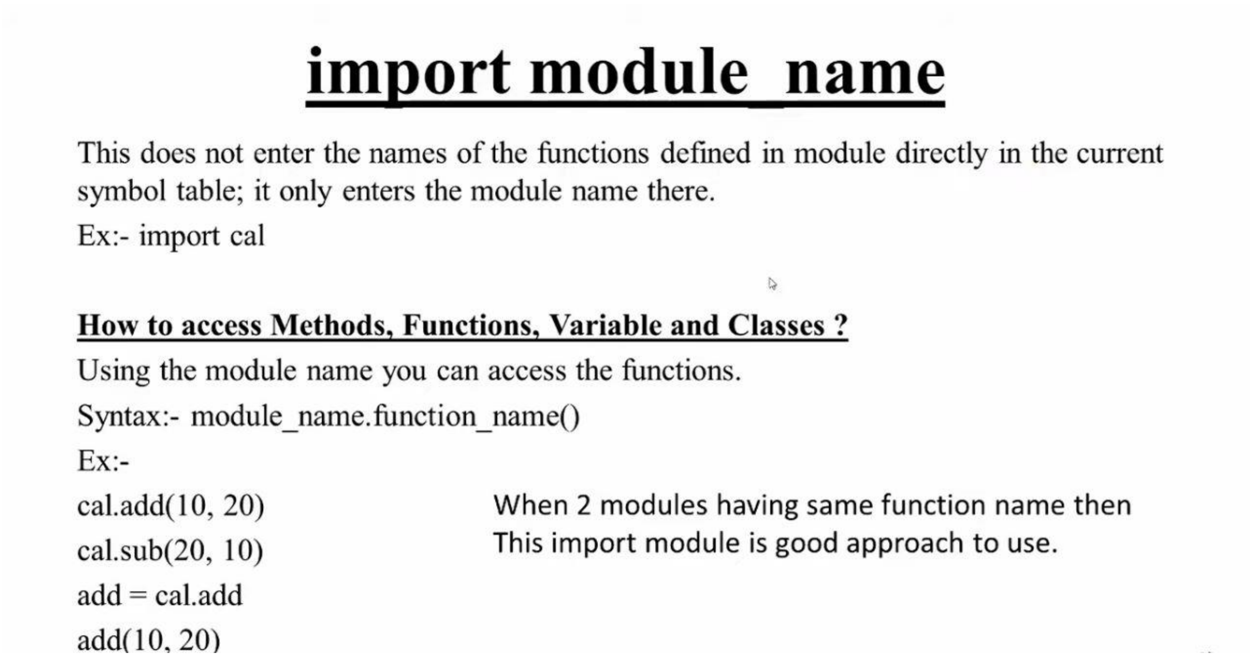
import module_name

import module_name as alias_name

from module_name import var_name, f_name, class_name, method_name……,

from module_name import f_name as alias_f_name

from module_name import *



import module_name

This does not enter the names of the functions defined in module directly in the current symbol table; it only enters the module name there.

Ex:- import cal

How to access Methods, Functions, Variable and Classes ?

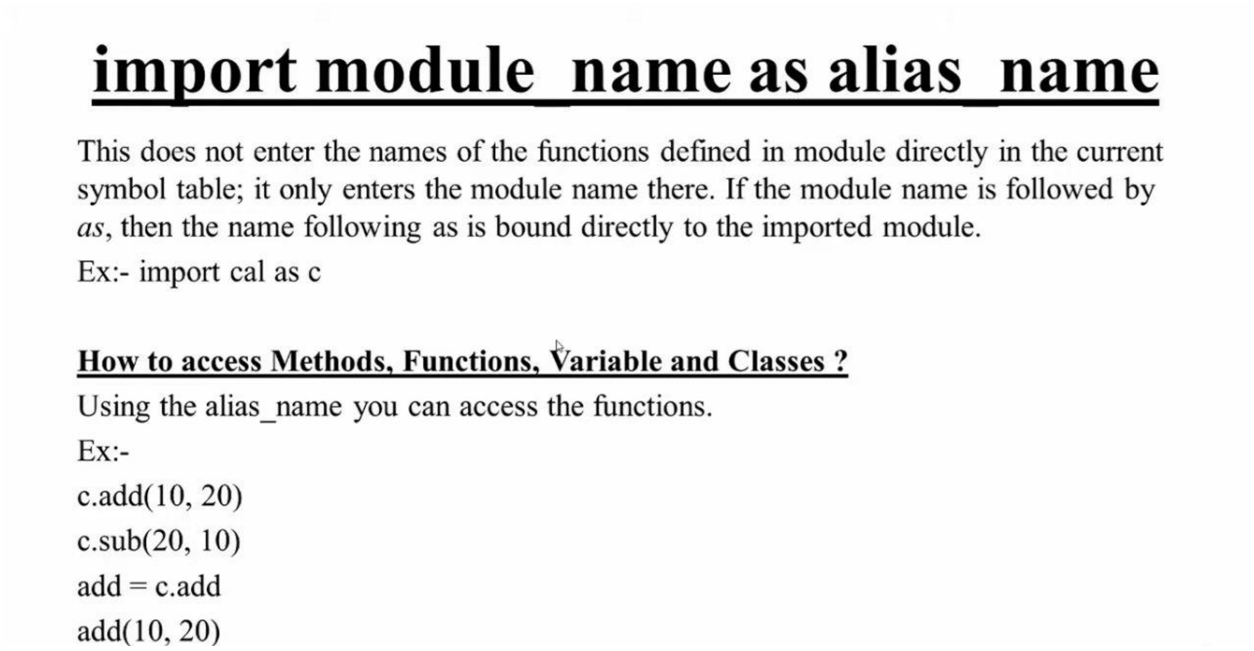
Using the module name you can access the functions.

Syntax:- module_name.function_name()

Ex:-

```
cal.add(10, 20)
cal.sub(20, 10)
add = cal.add
add(10, 20)
```

When 2 modules having same function name then This import module is good approach to use.



import module_name as alias_name

This does not enter the names of the functions defined in module directly in the current symbol table; it only enters the module name there. If the module name is followed by *as*, then the name following as is bound directly to the imported module.

Ex:- import cal as c

How to access Methods, Functions, Variable and Classes ?

Using the alias_name you can access the functions.

Ex:-

```
c.add(10, 20)
c.sub(20, 10)
add = c.add
add(10, 20)
```

from module_name import function_name

There is a variant of the import statement that imports names from a module directly into the importing module's symbol table.

Ex:- from cal import add, sub

How to access Methods, Functions, Variable and Classes ?

You can access the functions directly by it's name.

Ex:-

add(10, 20)

sub(20, 10)

from module_name import f_name as a_name

Ex:- from cal import add as s

How to access Methods, Functions, Variable and Classes ?

You can access the functions directly by it's alias name.

Ex:-

s(10, 20)

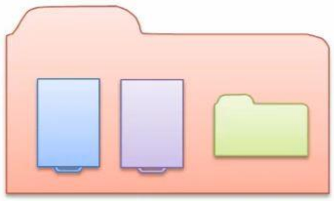
Package

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

A package can have one or more modules which means, a package is collection of modules and packages.

A package can contain packages.

Package is nothing but a Directory/Folder



from module_name import *

This imports all names except those beginning with an underscore (_).

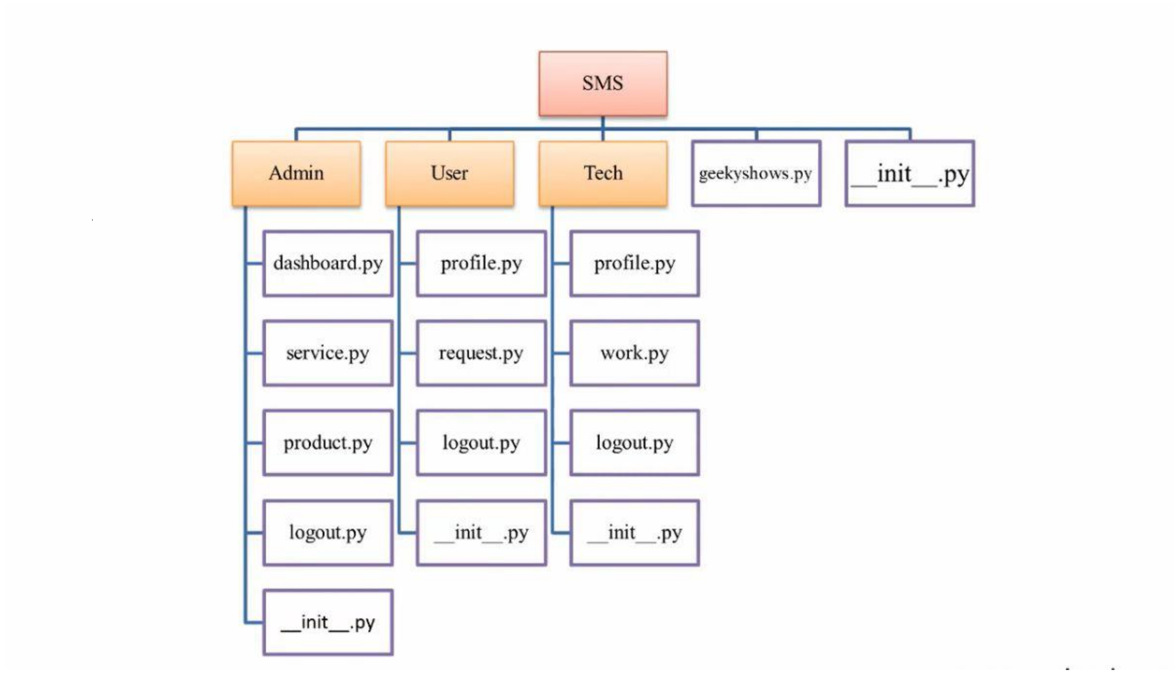
Ex:- from cal import *

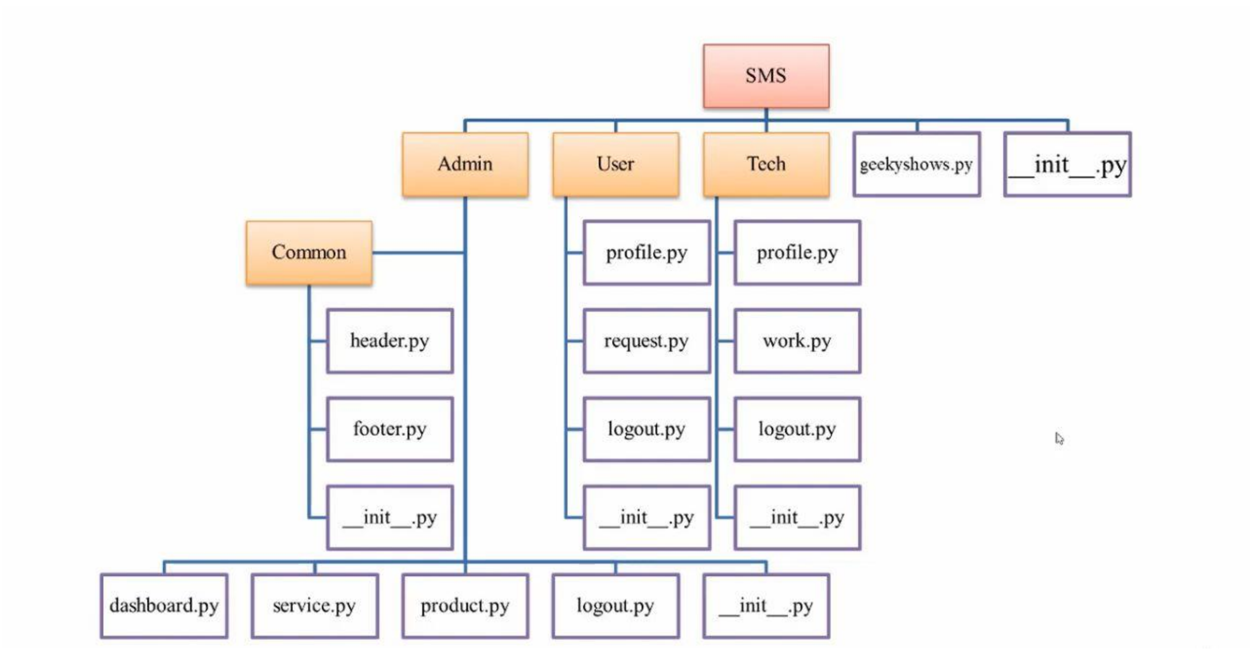
How to access Methods, Functions, Variable and Classes ?

You can access the functions directly by it's name.

Ex:-

```
add(10, 20)
sub(20, 10)
```


SMS
Admin
User
Tech
geekyshows.py
__init__.py
dashboard.py
service.py
product.py
logout.py
__init__.py
profile.py
request.py
logout.py
__init__.py
profile.py
work.py
logout.py
__init__.py


SMS
Admin
User
Tech
geekyshows.py
__init__.py
Common
header.py
footer.py
__init__.py
profile.py
request.py
logout.py
__init__.py
profile.py
work.py
logout.py
__init__.py
dashboard.py
service.py
product.py
logout.py
__init__.py

Creating Package

Package is nothing but a Directory/Folder which MUST contain a special file called *__init__.py*.

__init__.py file can be empty, it indicates that the directory it contains is a Python package, so it can be imported the same way a module can be imported.

How to use Package

Syntax:- import packageName**.**moduleName

Syntax:- import packageName**.**subPackageName**.**moduleName

Ex:- import Admin.service

Ex:- import Admin.Common.footer

How to Access Variable, Function, Method, Class etc. ?

Syntax:- packageName**.**moduleName.functionName()

Syntax:- packageName**.**subPackageName**.**moduleName.functionName()

Ex:- Admin.service.admin_service( )

Ex:- Admin.Common.footer.admin_common_footer( )