# SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

## Department of Artificial Intelligence and Data Science

## Course Name – Computational Thinking and Python Programming

## I Year / I Semester

### Unit 5-Files

## MODULES

Any file that contains Python code can be imported as a module. For example, suppose you have a file named wc.py with the following code:

```
def linecount(filename):
count = 0
for line in open(filename):
count += 1
return count
print linecount('wc.py')
```

If you run this program, it reads itself and prints the number of lines in the file, which is 7.
You can also import it like this:

```
>>>     import wc
7
```

Now you have a module object wc:

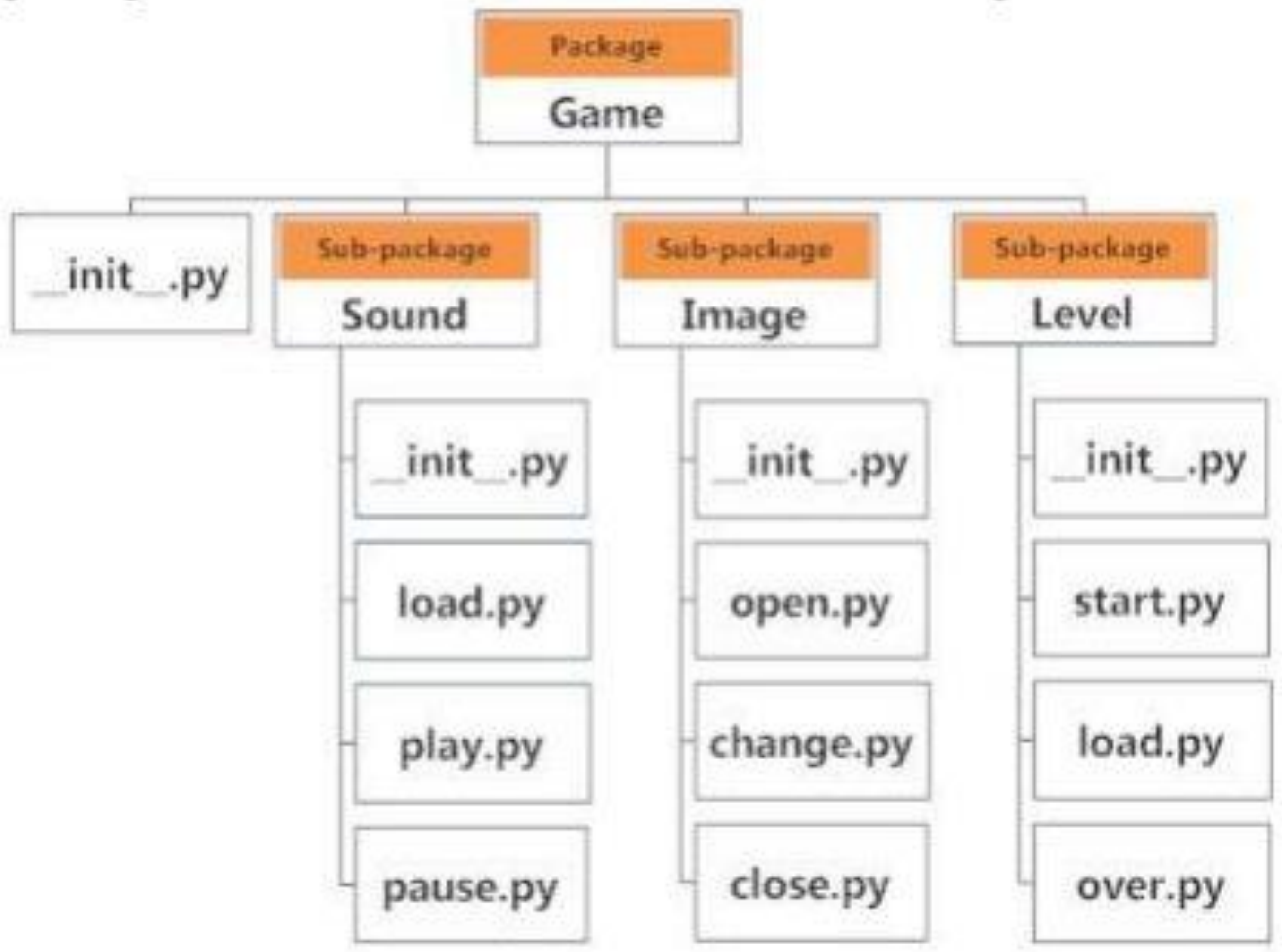
```
>>>     print wc
<module 'wc' from 'wc.py'>
>>>     wc.linecount('wc.py')
7
```

So that’s how you write modules in Python.
The only problem with this example is that when you import the module it executes the test code at the bottom. Normally when you import a module, it defines new functions but it doesn’t execute them.
Programs that will be imported as modules often use the following idiom: *if __name__ == '__main__':*
*print linecount('wc.py')*
__name__ is a built-in variable that is set when the program starts. If the program is running as a script, __name__ has the value __main__; in that case, the test code is executed. Otherwise, if the module is being imported, the test code is skipped.
Eg:

```
# import module import calendar

yy= 2017
mm = 8

# To ask month and year from the user
# yy = int(input("Enter year: "))
# mm = int(input("Enter month: "))

display the calendar
 print(calendar.month(yy, mm))
```

## PACKAGE

A **package** is a collection of modules. A Python package can have sub-packages and modules.
A directory must contain a file named __init__.py in order for Python to consider it as a package. This file can be left empty but we generally place the initialization code for that package in this file.
Here is an example. Suppose we are developing a game, one possible organization of packages and modules could be as shown in the figure below.



**Importing module from a package**

We can import modules from packages using the dot (.) operator.

For example, if want to import the start module in the above example, it is done as follows. *import Game.Level.start*
Now if this module contains a function named *select_difficulty(),* we must use the full name to reference it.
*Game.Level.start.select_difficulty(2)*

If this construct seems lengthy, we can import the module without the package prefix as follows.
*from Game.Level import start*

We can now call the function simply as follows.
*start.select_difficulty(2)*

Yet another way of importing just the required function (or class or variable) form a module within a package would be as follows.
*from Game.Level.start import select_difficulty*

Now we can directly call this function.
*select_difficulty(2)*

Although easier, this method is not recommended. Using the full namespace avoids confusion and prevents two same identifier names from colliding.
While importing packages, Python looks in the list of directories defined in sys.path, similar as for module search path.