Python Modules

What are modules in Python?

- Modules refer to a file containing Python statements and definitions.
- A file containing Python code, for example: `example.py`, is called a module, and its module name would be `example`.
- We use modules to break down large programs into small manageable and organized files. Furthermore, modules provide reusability of code.
- We can define our most used functions in a module and import it, instead of copying their definitions into different programs.

Let us create a module. Type the following and save it as `example.py`.

```python
# Python Module example

def add(a, b):
   """This program adds two
   numbers and return the result"""

   result = a + b
   return result
```

Here, we have defined a function `add()` inside a module named `example`. The function takes in two numbers and returns their sum.

How to import modules in Python?

We can import the definitions inside a module to another module or the interactive interpreter in Python.

We use the `import` keyword to do this. To import our previously defined module `example`, we type the following in the Python prompt.

```
>>> import example
```

This does not import the names of the functions defined in `example` directly in the current symbol table. It only imports the module name `example` there. Using the module name we can access the function using the dot `.` operator. For example:

```
>>> example.add(4,5.5)
9.5
```

Python import statement

We can import a module using the `import` statement and access the definitions inside it using the dot operator as described above. Here is an example.

```
# import statement example
# to import standard module math

import math
print("The value of pi is", math.pi)
```

When you run the program, the output will be:

```
The value of pi is 3.141592653589793
```

We can import a module by renaming it as follows:

```
# import module by renaming it

import math as m
print("The value of pi is", m.pi)
```

We have renamed the `math` module as `m`. This can save us typing time in some cases.

Python from...import statement

We can import specific names from a module without importing the module as a whole. Here is an example.

```
# import only pi from math module

from math import pi
print("The value of pi is", pi)
```

output:

The value of pi is 3.141592653589793

Import all names

We can import all names(definitions) from a module using the following construct:

```
# import all names from the standard module math

from math import *
print("The value of pi is", pi)
```

output:

The value of pi is 3.141592653589793

The dir() built-in function

We can use the `dir()` function to find out names that are defined inside a module.
For example, we have defined a function `add()` in the module `example` that we had in the beginning.
We can use `dir` in `example` module in the following way:

```
>>> dir(example)
['__builtins__',
'__cached__',
'__doc__',
'__file__',
'__initializing__',
'__loader__',
'__name__',
'__package__',
'add']
```

Here, we can see a sorted list of names (along with `add`). All other names that begin with an underscore are default Python attributes associated with the module (not user-defined).

Python Package

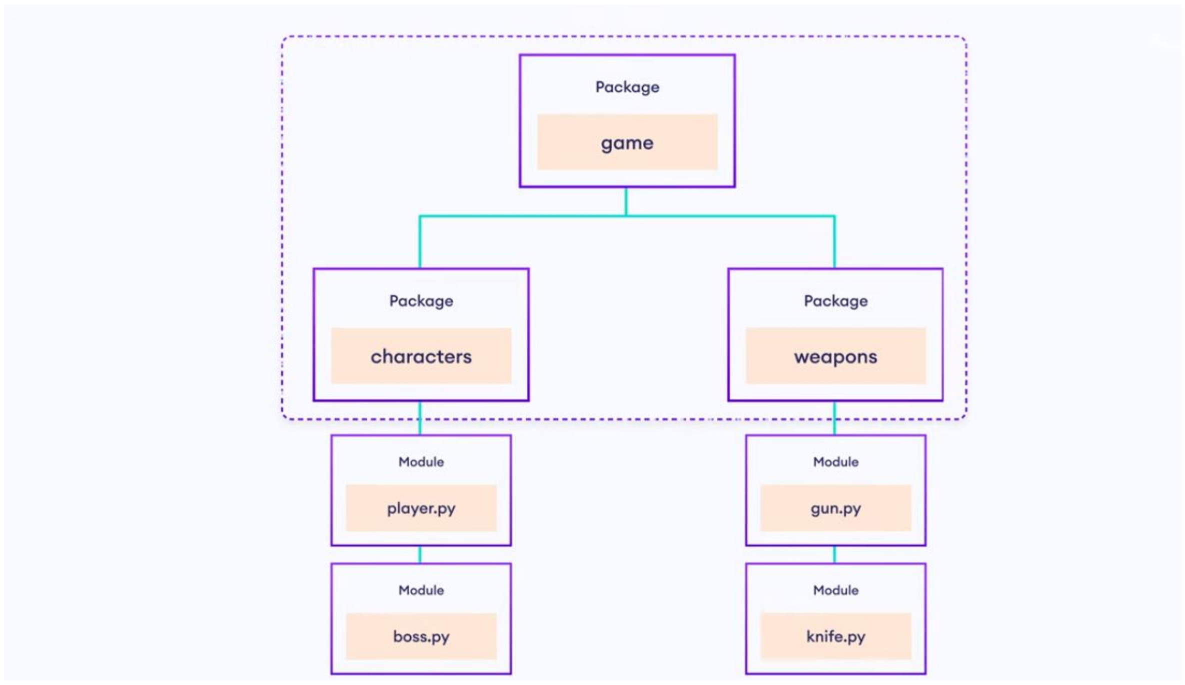
What are packages?

- We don't usually store all of our files on our computer in the same location. We use a well-organized hierarchy of directories for easier access.
- Similar files are kept in the same directory, for example, we may keep all the songs in the "**music**" directory. Analogous to this, Python has packages for directories and modules for files.
- As our application program grows larger in size with a lot of modules, we place similar modules in one package and different modules in different packages. This makes a project (program) easy to manage and conceptually clear.

Similarly, as a directory can contain subdirectories and files, 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.


Package
game
Package
characters
Package
weapons
Module
player.py
Module
gun.py
Module
boss.py
Module
knife.py