Python 101

Modules and Packages in Python

Contents

Source Code Encoding

We know that by default Python's source code files are encoded using UTF-8. But in order to change this encoding we can declare source code encoding as following -

```
# -*- coding: encoding -*-
```

eg.

- * - coding: cp1252 - * - ← By default in most Windows systems.

It is written always after the shebang.

Python Modules

Python modules are just a file in which the a bunch of code is written. In order to use the code written in that file in some other file or interpreter we can use it as **import** statement.

```
import module_name
```

see Day1/Code/ex1/script1.py

Python Modules contd..

Importing something specific from module

```
from module import something_specific1,
something_specific2
```

see Day1/code/ex1/script2.py

Importing everything from module

```
from module import *
```

Importing everything from module doesn't import methods starting with _.

Python Modules contd..

The module search path - Module search path defines the paths where our interpreter should try looking into in order to find the module.

By default it checks in our current diretory only **(.)**. If its not found then it checks for the module in the path given in **sys.path.**

Most often sys.path comes from PYTHONPATH named environment variable and also some installation dependent defaults.

Python Packages

Packages are a way of structuring Python's modules. They are just a bunch of modules. Its just a directory with a file named **__init__.py**

Whenever we import something from a package this file runs first.

see Day1/Code/ex2/script1.py

We can further create Packages inside packages that are called **subpackages.**

Mind that each subpackage must be a valid package. Ie. must contain an __init__.py

Python Packages contd..

Importing all from package - we can define what we want to export from a subpackage when all is imported from package using identifier **__all__** in **__init__.**

__all__ is a list of string_names of all of the modules that we want to export when * is imported from our package.

see Day1/Code/ex2/script2.py

Package imports are relative.

see Day1/Code/ex3/script1.py

Python Packages contd..

It is fairly possible that two packages might be having same modules or two modules might be having same method names which will clash with each other when imported into some other file for use.

In this case we use **as** keyword it creates an **alias** to the an object while importing.

eg. **from x import y as z** ← Now in our file y can be accessed using z.

see Day1/Code/ex3/script2.py

__name__ in Packages

__name__ contains the name of the module if imported somewhere and if it is directly ran in interpreter __name__ is given as __main__.