# Python Packages and Modules - Section A & B

## Section A: Basic Understanding

**1) What is a Python package? How is it different from a module?**

A Python package is a directory that contains multiple Python modules and a special file called \_\_init\_\_.py. A module is a single Python file, whereas a package can contain multiple modules and subpackages. Packages help organize and structure code hierarchically.

**2) What is the purpose of \_\_init\_\_.py in a package directory?**

The \_\_init\_\_.py file indicates that the directory is a Python package. It can also be used to execute initialization code and control what is exposed when the package is imported.

**3) What happens when you use from package import \* in Python?**

Using 'from package import \*' imports all public names defined in the package’s \_\_init\_\_.py file or in the modules listed in the \_\_all\_\_ list, if defined.

**4) What is the effect of defining \_\_all\_\_ in a package’s \_\_init\_\_.py file?**

The \_\_all\_\_ list defines the public interface of the package. Only the modules listed in \_\_all\_\_ will be imported when 'from package import \*' is used.

**5) How can you create and use a subpackage in Python?**

To create a subpackage, create a subdirectory inside a package and include an \_\_init\_\_.py file in it. Use dot notation to access modules in subpackages, e.g., 'from package.subpackage import module'

## Section B: Coding-Based Questions

**6) Create a package called math\_utils with the following modules:**

basic\_ops.py – containing functions for add, subtract  
advanced\_ops.py – containing functions for power and factorial  
Demonstrate how to import and use all functions using from math\_utils import \*.  
  
Directory structure:  
math\_utils/  
 \_\_init\_\_.py  
 basic\_ops.py  
 advanced\_ops.py  
  
basic\_ops.py:  
def add(a, b): return a + b  
def subtract(a, b): return a - b  
  
advanced\_ops.py:  
def power(a, b): return a \*\* b  
def factorial(n):  
 return 1 if n == 0 else n \* factorial(n - 1)  
  
Usage:  
from math\_utils import \*  
print(basic\_ops.add(2, 3))  
print(advanced\_ops.power(2, 3))

**7) Intra-package reference example:**

module\_a.py:  
from .module\_b import greet  
def call\_greet():  
 greet()  
  
module\_b.py:  
def greet():  
 print('Hello from module\_b')

**8) Import from subpackage using relative import:**

Directory structure:  
shapes/  
 \_\_init\_\_.py  
 area/  
 \_\_init\_\_.py  
 circle.py  
 rectangle.py  
  
circle.py:  
def area\_of\_circle(r):  
 return 3.14159 \* r \* r  
  
rectangle.py:  
from .circle import area\_of\_circle  
print(area\_of\_circle(5))

**9) Effect of defining \_\_all\_\_ in math\_utils \_\_init\_\_.py:**

\_\_init\_\_.py:  
\_\_all\_\_ = ['basic\_ops']  
  
Now, when you run:  
from math\_utils import \*  
Only basic\_ops module will be imported, and advanced\_ops will not be accessible unless imported explicitly.