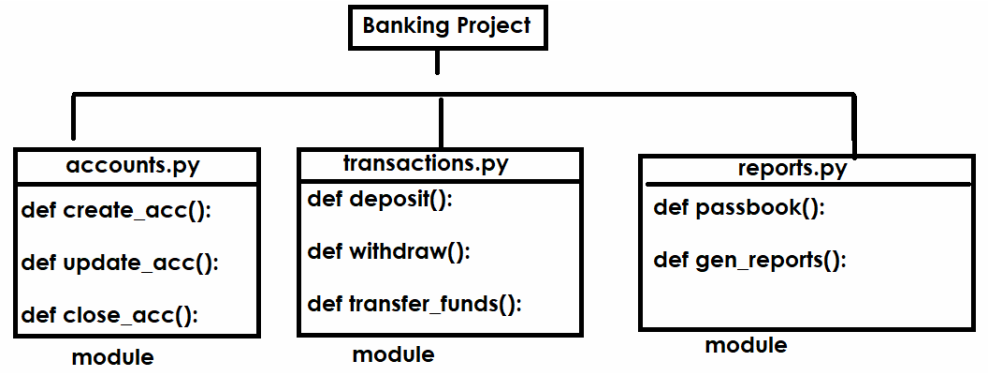
Modules and Packages

Python project/project is never developed by writing everything inside one program, it is divided into number programs (functionality of project is divided into number of programs).

Advantage of dividing into number of programs

1. Easy to maintain the code
2. Easy to understand
3. Simultaneously multiple people can develop project
4. Reusability between programs

What is module?
A module is nothing but a python program having extension .py
A module is collection of variables, functions and classes.



Python modules are 2 types

1. Predefined module
2. User defined module

Predefined module
The existing modules/programs are called predefined modules. These modules are provided by python or third party vendors.
Example: math.py, sys.py, datetime.py, random.py, calendar.py,…
These are called libraries.

User defined module
The programs developed by programmer are called user defined programs or modules. These are application specific.
Example: accounts.py, transactions.py, reports.py

User defined modules are two types

1. Executable module
2. Non executable module

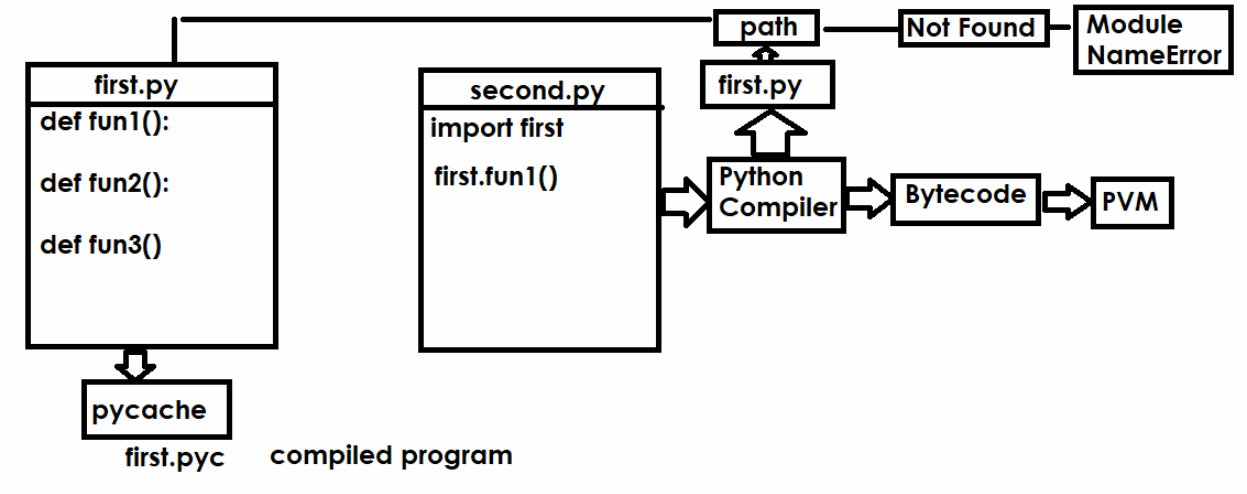
Executable module
A module or program which consists of executable statements is called executable module, this module generates output.

Non Executable module
A module or program which does not have executable statements is called non executable module. It is a reusable module; the functionality of this module is used inside other modules.

How to use the content of one module inside another module?
The content of one module can be used inside another using "**import"** keyword.

Syntax1: import module-name
Syntax2: import module-name as alias-name
Syntax3: from module-name import *
Syntax4: from module-name import identifier,identifier,..
Syntax5: from module-name import identifier as alias-name
Syntax6: import module-name,module-name

Importing is not including module, importing is verifying/identifying module and module content. The content of module is included or imported during runtime.



Python uses dynamic linking or loading. Linking of modules is done at runtime.

How to compile user defined modules/programs explicitly?

```
D:\fspmar5pm>python -m py_compile test297.py

D:\fspmar5pm>python -m py_compile test.py

D:\fspmar5pm>python --version
Python 3.12.6

D:\fspmar5pm>python -m py_compile test.py

D:\fspmar5pm>
```

What is __pycache__?

The __pycache__ directory in Python is used to store the bytecode-compiled versions of Python files. When a Python script is executed or a module is imported, the interpreter compiles the source code into bytecode, which is a lower-level, platform-independent representation of the

code. This compilation process is done to optimize the execution speed for subsequent runs.

Test298.py (reusable module)	Test299.py (executable module)

def fun1():
  print("this is function1")
def fun2():
  print("this is function2")
def fun3():
  print("this is function3")
def fun4():
  print("this is function4")
def fun5():
  print("this is function5")
def fun6():
  print("this is function6")
	import test298
	
	test298.fun1()
	test298.fun2()
	test298.fun3()
	
	Output
	this is function1
	this is function2
	this is function3

What is namespace in python?
A namespace in Python is a system that provides a unique name to every object in Python. It is implemented as a dictionary that maps names to objects. Namespaces help to avoid naming conflicts, ensuring that each name is unique.

Python created different name spaces
Built-in namespace: Contains built-in functions and exception classes available in Python.
Global namespace: Contains objects defined at the top level of a module or script.
Local namespace: Created when a function is called and contains objects defined within the function.
Module namespace: Contains objects defined within a module.
Class namespace: Contains objects defined within a class.