modules

- A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.
- Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.
- Packages in Python
- A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and subpackages and sub-subpackages, and so on.
- Consider a file *Pots.py* available in *Phone* directory.

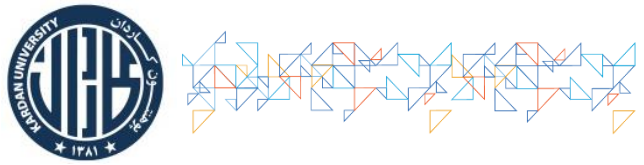
Modules

- Python Modules are organizational units which packages program's code and data for reuse.
- Each file is a module
- Modules import other modules to use the names they define.
- Python's modules allow us to link individual files into a larger program system.
- We can Reuse the modules in python using :
- import
- from

Modular Programming

- **Modular programming** refers to the process of breaking a large, unwieldy programming task into separate, smaller, more manageable subtasks or **modules**.

- **Simplicity:**
- **Maintainability:**
- **Reusability:**



Why Modules?

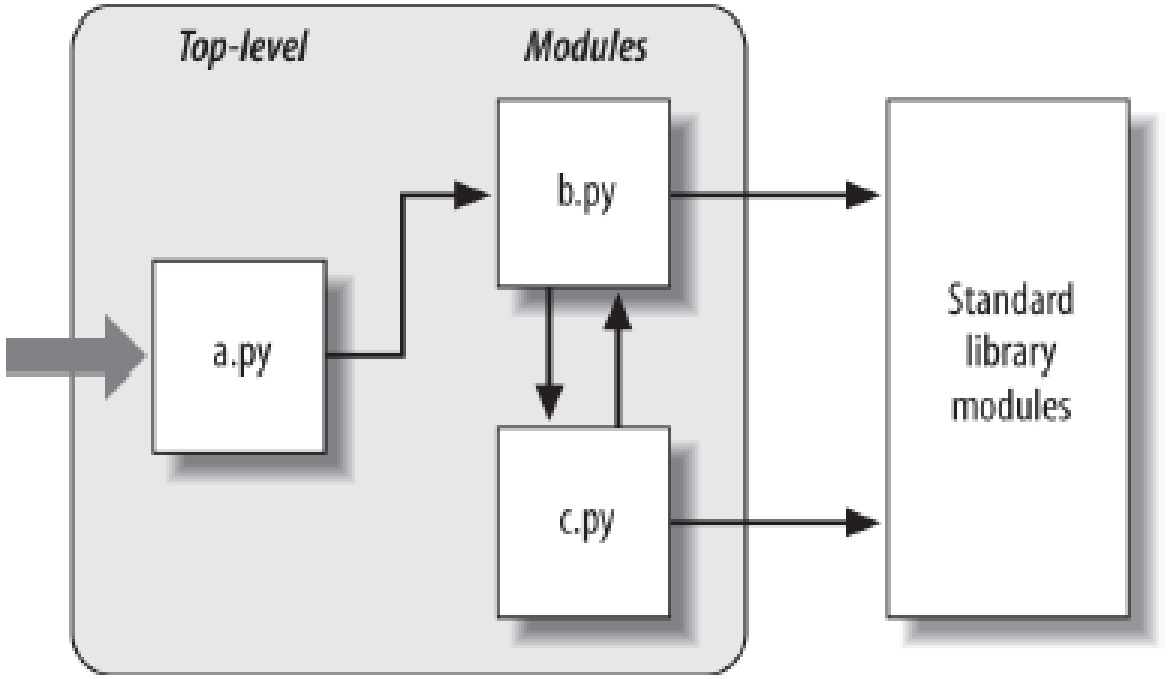
Modules provide an easy way to organize components into a system by serving as self-contained packages of variables known as *namespaces*.

- All the names defined at the top level of a module file become attributes of the imported module object.
- A module serve to do the following:
- *Code reuse*
  - This supports a *modular* program design that groups functionality into reusable units.
- *System namespace partitioning*
- *Implementing shared services or data*

How to use modules

- # b.py file
- import c

- #a.py file
- import b



Create Modules

- # Module1.py

```
1  #module1.py
2
3  def myFunction(test):
4      print ('This is the text: ', test)
5
```

```
1  #module2
2  import module1
3
4  module1.myFunction("First Module")
5
```

```
This is the text:  First Module
```

From and import

- Import is used to fetch the module as a whole

  import module1

From is used to get the copy of some function.

  from module1 import myFunction

- Are changes happening in the modules?

module2.py

```
from small import x, y          # Copy two names out
x = 42                          # Changes local x only
y[0] = 42                       # Changes shared mutable in place
```

small.py

```
x = 1
y = [1, 2]
```

x=42

y= [42,2]

What import does?

- It perform three distinct steps the first time a program imports a given file:

- 1. *Find* the module's file.
- 2. *Compile* it to byte code (if needed).
- 3. *Run* the module's code to build the objects it defines.

Usage of import and from

- **import** assigns an entire module object to a single name.
- **from** assigns one or more names to objects of the same names in another module.

- from statement may fail to work in some cases
- Suppose two modules with the same name
- A.py
- def func():
  - print(“a”)

B.py

```
def func():
    print(“b”)
```

c.py

```
from A import *
from B import *
func()
#  ….. Only the last one
```

Reloading Modules

- Imports (via both import and from statements) load and run a module's code only the first time the module is imported in a process.
- it always uses the first version of the modules we are developing, even we have made changes to the code.
- The reload function forces an already loaded module's code to be reloaded and rerun.
- We will achieve dynamic customization using reload
- reload is a function in Python,
- reload(module)

Use Imports

```
#module a.py
X="First Version"
```

```
1  import a
2  print(a.X)
```

```
First Version
```

```
#module a.py
X="Second Version"
```

```
1  import a
2  print(a.X)
3
4  import a
5  print(a.X)
6
7  from imp import reload
8  reload(a)
9  print(a.X)
10
```

```
First Version
First Version
Second Version
```

```
1  import a
2  print(a.X)
3
4  import a
5  print(a.X)
6
7  from imp import reload
8  reload(a)
9  print(a.X)
10
```

```
First Version
First Version
Second Version
```

Name spaces

- dir() / vars(module) function provides the namespaces available for a module

- These namespaces can contain all the variables, methods and classes available in python program

Package imports

- If you have multiple files in folders and you want to import them in your program, you need to add the folder in the search path directory:
- Dir1/a.py
- import Dir1.a

- Each directory named within the path of a package import statement must contain a file named *__init__.py*,