



Modules

Agenda

1

What is a module?

2

How to create and import a module?

3

Different ways of importing a module

4

How does import searches for the modules?

5

Built in modules

What is a module?



What is a module?

- Python provides a way to put definitions in a file and use them in another scripts. Such a file is called a module.
- **A module is a .py file containing Python definitions and statements.**
- The file name is the module name.
- Modules helps us in splitting longer programs into several files for easier maintenance.
- Modules helps us in using a handy function that we have written, in several other programs without copying its definition into each program.
- Grouping related code into a module makes the code easier to understand and use.
- **A module allows you to logically organize your Python code.**

How to create and import a module?



Create a module

- A module is a .py file containing definitions and statements.
- File name is the module name.

Let us create a calculator module which contains definitions for performing the basic operations such as add, sub, div and mul for two inputs:

Define these methods:

```
def add(x,y):  
    return(x+y)
```

```
def sub(x,y):  
    return(x-y)
```

```
def mul(x,y):  
    return(x*y)
```

```
def div(x,y):  
    return(x//y)
```

How do we save this module?

Answer: calculator.py

Import a module

- Modules are imported in another script with the help of **import** statements.

Basic syntax: `import module_name`

- It is recommended to place all the import statements at the beginning of our script.
- Importing the calculator module:

`import calculator`

What does import statement do?

- It searches for the mentioned module and loads it.
- Now all the definitions and statements in that module are available for us.
- A module is loaded only once, regardless of the number of times it is imported.

Import a module

How to call the functions present in the module?

- Basic syntax:

```
module_name.function_name(optional_arguments)
```

- Calling the add function from calculator module:

```
demo.py
```

```
import calculator
```

```
result = calculator.add(5,5)
```

```
print(result)
```


Import a module : Example

calculator.py

```
def add(x,y):  
    return(x+y)
```

```
def sub(x,y):  
    return(x-y)
```

```
def mul(x,y):  
    return(x*y)
```

```
def div(x,y):  
    return(x//y)
```

demo.py

```
import calculator
```

```
print('Add:',calculator.add(5,5)) #Add: 10
```

```
print('Sub:',calculator.sub(10,5)) #Sub: 5
```

```
print('Mul:',calculator.mul(3,2)) #Mul: 6
```

```
print('Div:',calculator.div(10,2)) #Div: 5
```

Different ways of importing a module



Different ways of importing a module

Importing a specific function from a module:

- Let us assume we have a module support.py which contains three functions in it: check(), calculate(x, y) and remove(). We need only calculate(x, y).

Basic syntax: `from module_name import function_name`

- When we import using from keyword, we can use the function directly in our script.
- Prefixing the module name is not required to use that function.

```
from support import calculate  
calculate(10, 20)
```

Different ways of importing a module

Importing specific functions from a module:

Basic syntax:

```
from module_name import function1_name, function2_name,.....
```

Example:

```
from support import calculate, check
```

Importing a specific variable from a module:

Basic syntax: *from* module_name *import* variable_name

Example: *from* sys *import* argv

Different ways of importing a module

Importing a module with alias:

- We can import a module(function/variable) with a different name using **as** keyword.

Basic syntax:

```
import module_name as alias_name
```

```
from module_name import function_name as alias_name
```

```
from module_name import variable_name as alias_name
```

Example:

```
from support import calculate as calc
```

```
calc(15,20)
```

How does import searches for the modules?



How does import searches for the modules?

- When a module named spam is imported, the interpreter first searches for a built-in module with that name.
- If not found, it then searches for a file named spam.py in a list of directories given by the variable `sys.path`.

`sys.path` is initialized from these locations:

- The current directory containing the input script.
- `PYTHONPATH` (a list of directory names).
- The installation-dependent default.

Built in modules



Built in modules

- In addition to built-in functions, a large number of pre-defined functions are also available as a part of Python libraries. These functions are defined in modules.
- We can get a list of all available modules by running this command `help('modules')`

Sample:

```
docs          nose          sys
doctest       ntpath         sysconfig
draw          nturl2path     sysfont
draw_py       numbers        sysfont_test
draw_test     numexpr        syslog
dummy_threading numpy           tables
easy_install  opcode         tabnanny
email         operator       tarfile
encodings     optparse       telnetlib
enum          os             tempfile
```

Built in modules

Some of the important built in modules we use in day to day programming:

Module name	Use
os	It provides functions to work with directories.
sys	It provides functions and variables to manipulate different parts of the Python runtime environment.
math	It provides important mathematical functions like trigonometric, logarithmic etc..
random	It provides useful pseudo-random number generator functions.

Built in modules

Example for using math module:

```
main.py
1  import math
2
3  print(math.pi)
4  print(math.pow(2,4))
5  print(math.sqrt(100))
```

3.141592653589793
16.0
10.0

Built in modules

Example for using random module:

```
main.py
1 import random
2 #Returns a random integer between
3 #the specified integers.
4 num = random.randint(1,50)
5 print("Random number",num)
6
7 #Returns a randomly selected element
8 #from a non-empty sequence.
9 ch = random.choice('Laptop')
10 print("Random element",ch) |
```

Random number 27
Random element p



Thank you