Python Modules

Definition

- ❑A module is a Python object with arbitrarily named attributes that can be bind and reference.
- ❑Modules refer to a file containing Python statements and definitions.
- ❑Grouping related code into a module makes the code easier to understand and use.
- ❑Modules are used to break down large programs into small manageable and organized files.
- ❑A module can define functions, classes and variables
- ❑Furthermore, modules provide reusability of code.

Module Continue

Let us create a module. Type the following and save it as example.py.

```
def add(a, b):
   """This program adds two
   numbers and return the result"""

   result = a + b
   return result
```

- Here, we have defined a function add() inside a module named example. The function takes in two numbers and returns their sum.

Importing Module in Python

- ❑We can import the definitions inside a module to another module or the interactive interpreter in Python.
- ❑We use the import keyword to do this.
- ❑To import our previously defined module example, we type the following in the Python prompt.

```
>>> import example
```

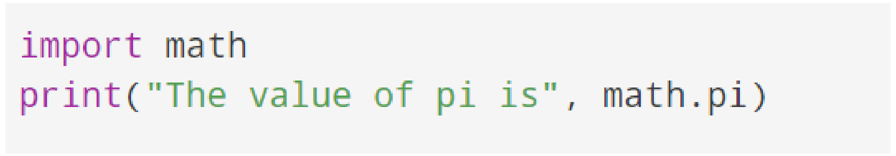
- ❑This does not import the names of the functions defined in example directly in the current symbol table. It only imports the module name example there.
- ❑Using the module name we can access the function using the dot . operator.

Importing Module in Python

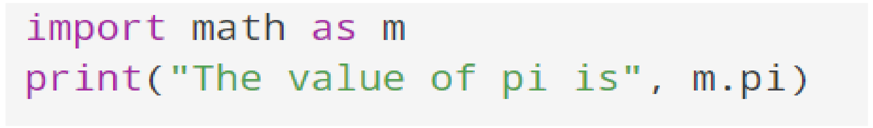
- ❑Python has tons of standard modules.
- ❑The full list of Python standard modules and their use cases can shown using the dir().
- ❑These files are in the Lib directory inside the location where you installed Python.
- ❑Standard modules can be imported the same way as we import our user-defined modules.
- ❑There are various ways to import modules.

Python Import Statement

- A module can be import using the import statement and access the definitions inside it using the dot operator.



```python
import math
print("The value of pi is", math.pi)
```

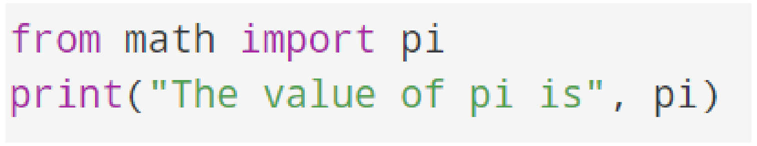


```python
import math as m
print("The value of pi is", m.pi)
```

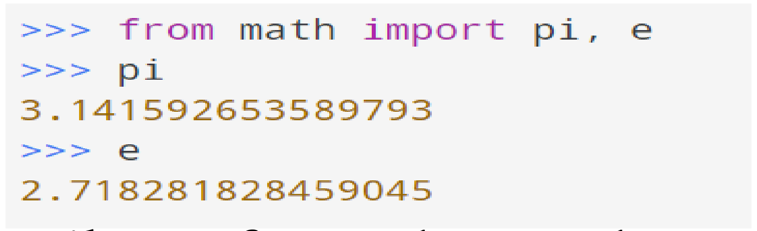
- We have renamed the math module as m. This can save us typing time in some cases.
- Note that the name math is not recognized in our scope. Hence, math.pi is invalid, and m.pi is the correct implementation.

Python from...import statement

- ❑Import specific names from a module without importing the module as a whole. Here is an example.



```python
from math import pi
print("The value of pi is", pi)
```



```
>>> from math import pi, e
>>> pi
3.141592653589793
>>> e
2.718281828459045
```

- ❑Imported only the pi attribute from the math module.
- ❑We can also import multiple attributes as follows:

Import all names

❑All names(definitions) from a module can be imported using the following construct:

```
from math import *
print("The value of pi is", pi)
```

❑All the definitions from the math module have been imported.

❑This includes all names visible in our scope except those beginning with an underscore(private definitions).

❑Importing everything with the asterisk (*) symbol is not a good programming practice.

❑This can lead to duplicate definitions for an identifier.

❑It also hampers the readability of our code.

Python Module Search Path

- ❑While importing a module, Python looks at several places.
- ❑ Interpreter first looks for a built-in module. Then(if built-in module not found),
- ❑Python looks into a list of directories defined in sys.path.
- ❑The search is in this order.
  - ❑The current directory.
  - ❑PYTHONPATH (an environment variable with a list of directories).
  - ❑The installation-dependent default directory.

Reloading a module

- The Python interpreter imports a module only once during a session.
- This makes things more efficient.
- Here is an example to show how this works.
- Suppose we have the following code in a module named my_module.

```
print("This code got executed")
```

- Now we see the effect of multiple imports.

```
>>> import my_module
This code got executed
>>> import my_module
>>> import my_module
```

The dir() built-in function

- The dir() function can be used to find out names that are defined inside a module.
- For example, the function add() in the module example as been defined at the beginning.
- The dir can be used in example module in the following way:
- >>>dir(example)
- Here, a sorted list of names (along with add) are shown.
- All other names that begin with an underscore are default Python attributes associated with the module (not user-defined).

Quiz

- Write a python module that will perform two basic functions of:
  - Addition of two numbers
  - Subtraction of two number
- Write a python code that will allow you call the add function from the module.