Modules

Modules refer to a file containing Python statements and definitions.

A file containing Python code, for e.g.: abc.py, is called a module and its module name would be "abc".

We use modules to break down large programs into small manageable and organized files. Furthermore, modules provide reusability of code.

We can define our most used functions in a module and import it, instead of copying their definitions into different programs.

How to import a module?

We use the import keyword to do this.

In [2]:
```
import example #imported example module
```

Using the module name we can access the function using dot (.) operation.

In [3]:
```
example.add(10, 20)
```

Out[3]:
```
30
```

Python has a lot of standard modules available.

https://docs.python.org/3/py-modindex.html (https://docs.python.org/3/py-modindex.html)

Examples:

In [5]:
```
import math
print(math.pi)
```

```
3.141592653589793
```

In [6]:
```
import datetime
datetime.datetime.now()
```

Out[6]:
```
datetime.datetime(2017, 10, 18, 20, 47, 20, 606228)
```

import with renaming

In [8]:
```
import math as m
print(m.pi)
```

```
3.141592653589793
```

from...import statement

We can import specific names form a module without importing the module as a whole.

In [9]:
```
from datetime import datetime
datetime.now()
```

Out[9]:
```
datetime.datetime(2017, 10, 18, 20, 47, 38, 17242)
```

import all names

In [10]:
```
from math import *
print("Value of PI is " + str(pi))
```

```
Value of PI is 3.141592653589793
```

dir() built in function

We can use the dir() function to find out names that are defined inside a module.

In [11]:
```
dir(example)
```

Out[11]:
```
['__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'add']
```

In [13]:
```
print(example.add.__doc__)
```

```
    This program adds two numbers and return the result
```