Programing in Python

Lecture 8 - Modules

Instructor: Zhandos Yessenbayev

Outline

- Program Flow
- Modular Programming
- Python Modules
- Built-in Modules

Program Flow

- Programming patterns we saw:
    - Sequential code
    - Conditional code (if statements)
    - Repetitive code (loops)
    - Store and reuse (functions)

Program Flow

```
try:
    fhand = open("input.txt")
except:
    print('File cannot be opened')
    exit()

def greet():
    print('Hello, world!’)

def bye():
    print('Good bye!')

for line in fhand:
    line = line.rstrip()
    if line.startswith('Hello'):
        greet()
    else:
        bye()
fhand.close()
```

Conditionals

Functions

Loops

Conditionals

Sequential

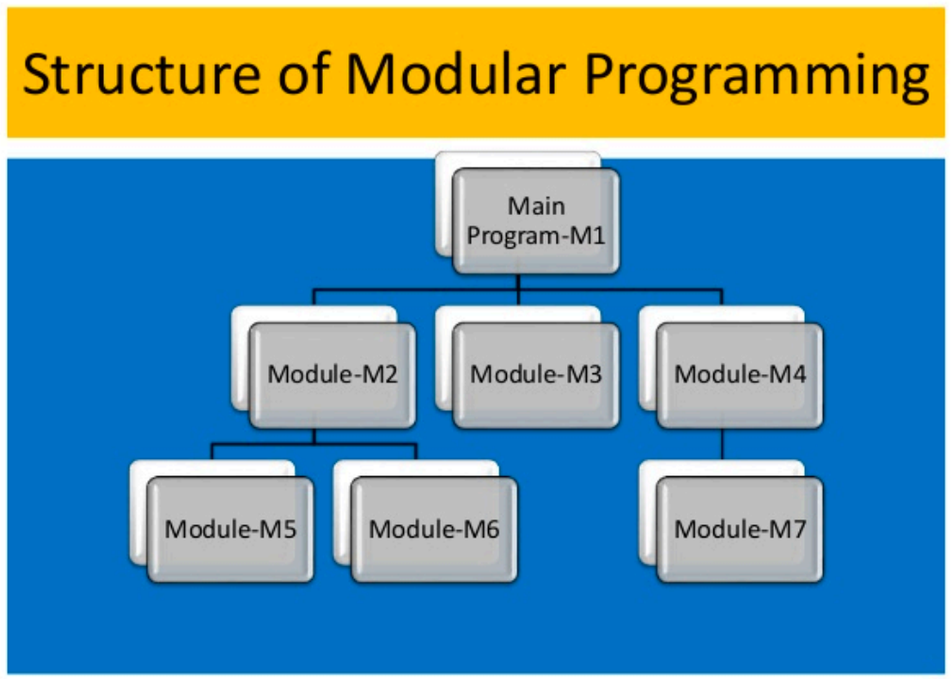
Modular Programming

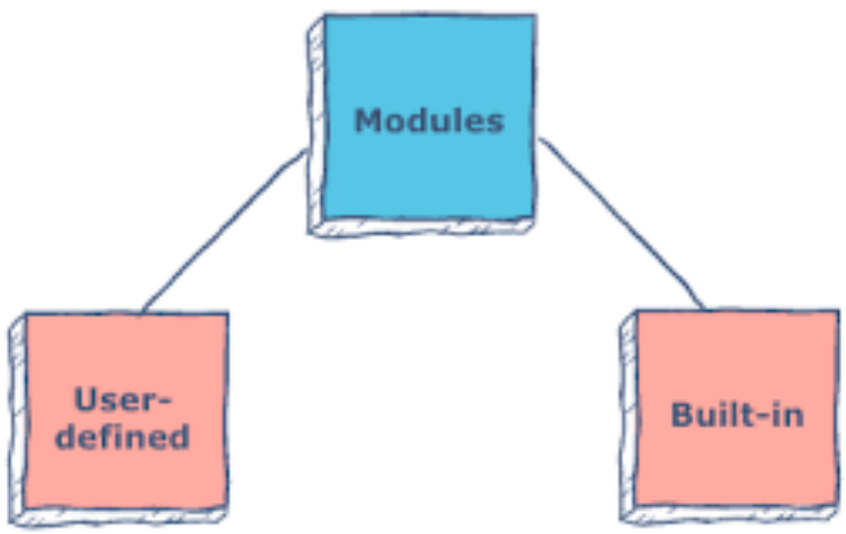
- **Modular programming** refers to the process of breaking a large task (program) into separate, smaller, more manageable subtasks or modules.

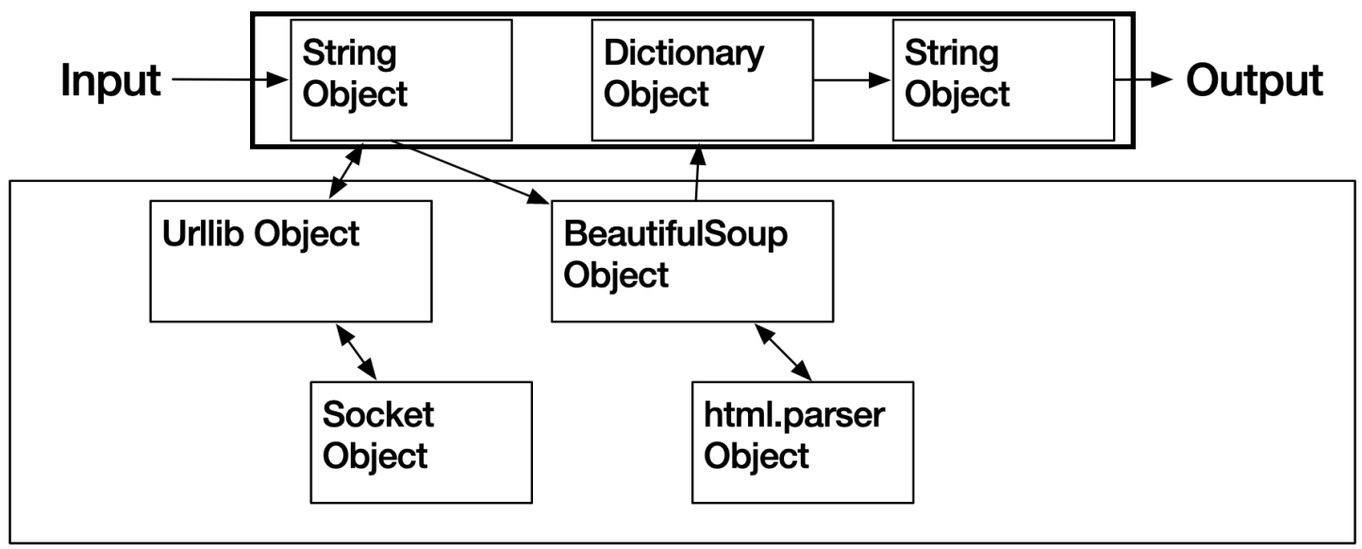
- Advantages of modular programming:
  - Simplicity - simpler to work on smaller tasks
  - Maintainability - easy to modify code
  - Reusability - modules can be used in another projects
  - Scoping - avoid collisions between names (variables, functions)

Modular Programming

- Python allows three way to define modules:
  - User-defined modules (*.py)
  - Built-in modules (math, sys)
  - [Third-party modules]







Python Modules

- Python module is a normal Python program with ***.py** extension

mymodule.py

```
pi = 3.14

def area(r):
    P = pi * r * r
    return P
```

main.py

```
import mymodule

rad = input("Enter circle radius: ")

A = mymodule.area(float(rad))

print("Circle's area:", A)
print("Pi:", mymodule.pi)
```

Syntax for importing a module:

```
import <module_name>
```

Naming Modules

- We can *rename* module's name in the program

mymodule.py

```
pi = 3.14

def area(r):
    P = pi * r * r
    return P
```

```
>>> import mymodule as mod
>>>
>>> A = mod.area(5)
>>> print(A)
>>> print(mod.pi)
```

Importing *from* Modules

- We can import some functions or variable ***from*** module

mymodule.py

Syntax for importing a module:

```
from <module_name> import <name(s)>
```

```
>>> from mymodule import area, pi
>>>
>>> A = area(5)
>>> print(A)
>>> print(pi)
```

```
>>> from mymodule import *
>>>
>>> dir()
```

Running Modules

- We can *run* modules as a Python program

mymodule.py

```
pi = 3.14

def area(r):
    P = pi * r * r
    return P

print(area(5))
```

Test the function, for debugging

```
$ python3 mymodule.py
$
$ python3 main.py
$
```

What are the outputs?

Running Modules

- We can *run* modules as a Python program

mymodule.py

```
pi = 3.14

def area(r):
    P = pi * r * r
    return P

if __name__ == "__main__":
   print(area(5))
```

Check if not loaded as a module

```
$ python3 mymodule.py
$
$ python3 main.py
$
```

Notice the difference

Built-in Modules

- Python has many useful **built-in** modules:
  - math, sys, os, string, html, random, time …
  - https://docs.python.org/3.8/py-modindex.html

```
>>> help('modules')
>>>
```

math Module

- Built-in **math** module has a set of mathematical methods and constants: https://docs.python.org/3.8/library/math.html

```
>>> import math
>>>
>>> pi = math.pi
>>> print(pi)
>>>
>>> x = math.sqrt(9)
>>> print(x)
>>>
>>> a = math.floor(2.4)
>>> b = math.ceil(2.4)
>>> print(a, b)
>>>
```

os Module

- os module provides way of using operating system dependent functionality.
https://docs.python.org/3.8/library/os.html#module-os

```
>>> import os
>>>
>>> os.getcwd()   # get current working directory
>>>
>>> os.mkdir(“newdir”)  # create new directory
>>>
>>> os.chdir("newdir") # changing to new directory
>>> os.chdir(“..")
>>>
>>> os.listdir()  # list the files and directories
>>>
>>> os.rmdir(“newdir") # remove the new directory
>>>
```

sys Module

- **sys** module provides functions and variables used to manipulate different parts of the Python runtime environment
  https://docs.python.org/3.8/library/sys.html#module-sys

```
>>> import sys
>>>
>>> sys.path # lists the search path for modules
>>>
>>> sys.version
>>>
```

```
import sys

print("You entered: “, sys.argv[0], sys.argv[1])
```

Thanks!