Module and Package

Python Modules

A library is a collection of code

- contain commonly used functionality in the form of function (and objects)
- makes coding more efficient

There are many built-in libraries, known as modules that come with Python

Examples: `time, random`

- which contain functions that you can use in your program

To use the functions provided by the modules, use the "`import`" keyword:

```
import time
import random
```

- which load all the functions in each module into the memory

To get the list of functions (and variables) available in a module:

```
dir(random)
```

Python Modules

You can use the specific function in the module after importing it.

```
import random
for _ in range(10):
    color = random.choice((“R", “G", “B“))
print(color)
```

Instead of importing all the functions in the module

- you can import only the specific function that you want
  - and also give it a different (shorter) name
- which use less memory (in your computer)

```
from random import choice as ch
for  _ in range(10):
    print(ch(["red", "green", "blue"]))
```

Creating own Module

You can also create your own module to store your own functions

- such as when your program size become too big to manage easily in one file

A custom module is a .py file that contains functions definition

(i) Create a module contains the following function and save as **my_module.py**

```
def greeting1(name):
    print(f"Hello, {name} !")

def greeting2(name):
    print(f"Good Day, {name} !")
```

(ii) In your program file **my_prog.py**

```
import my_module
mymodule.greeting1("Nick")
mymodule.greeting2("Alex")
```

Testing a Module

Typically, we will also want to test run the module separately

- without writing another program to import it

```
def main():
    greeting1("test1")
    greeting2("test2")

def greeting1(name):
    print(f"Hello, {name} !")

def greeting2(name):
    print(f"Good Day, {name} !")

if __name__ == "__main__": # will only be equal when this
    main()                  # py file is run on its own
```

Regular Expression Module

Python has a built-in module, **re**, known as Regular Expression

- which can be used to search for a sequence of characters of specific pattern
  - return 'None' if not matched
- useful for searching whether a string contains certain pattern

Example:

```
import re
text = "That cloth is very nice"
match = re.search("^That.*nice$", text)
if match:
   print("matched!")
```

where

```
^   starts with
$   ends with
.   any character
*   zero or more occurrences
[]  a set of character
```

Finger Exercise

Code a function `strong_password()` that will prompt the user to select a strong password.

```
def strong_password():
```

Package and Library

There are 3rd party code that are available for us to use in our programme

- available as files known as **Package** and **Library**
  - a package contains collection of modules (e.g. numpy, pandas)
  - a library contains collections of packages (e.g. matplotlib)
- may contain code written in other language (e.g. c for optimization purposes)

To use the modules in these packages, we can use the PIP package manager with the command `pip install` (in the text based command prompt window)
(For our case with miniconda, we can also use `conda install` command, or through Anaconda Navigator if you have installed it)

Example: to install the package `pandas`, open a command window and types

```
pip install pandas
```

Once it is installed, include `import pandas` in your program to use its functions

Aside: Conda Install

For package installation using conda install command

- by default, it will go to Anaconda's default site, known as 'Channels' to download the package (which is free for non-commercial use)

```
(base) C:\Users\aschvun>conda install pandas
3 channel Terms of Service accepted
Retrieving notices: done
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
```

But we can also specify other channel where conda will try to find the package first

- such as the popular community-driven channel conda-forge.

```
(base) C:\Users\aschvun>conda install -c conda-forge pandas
3 channel Terms of Service accepted
Retrieving notices: done
Channels:
 - conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): | |
```