모듈(Module)

- 독립적인 기능을 가지고 재사용가능한 프로그램 단위를 모듈이라고 한다.
- **파이썬에서 모듈**은 재사용 가능한 함수, 클래스등을 작성한 소스 파일을 말한다.
  - 함수나 클래스를 작성한 `.py` 스크립트 파일 파일이 모듈이 된다.
- 모듈의 함수나 클래스들을 다른 python 프로그램에서 호출 하여 사용할 수 있다.
  - 단 사용하기 위해서는 `import` 를 먼저 해야 한다.
- 이런 모듈들을 모아 놓으면 라이브러리가 된다.
- **모듈의 종류**
  - 표준 모듈
    - 파이썬에 내장된 모듈
  - 사용자 정의 모듈
    - 개발자가 재사용을 위해 직접 만든 모듈
  - 3rd Party 모듈
    - 특정 개발업체나 개발자들이 만들어 배포하는 모듈
    - 사용자 정의 모듈도 배포되어 다른 곳에서 사용되면 3rd party 모듈이 된다.

패키지 (Package)

- 모듈들을 모아 놓은 것을 패키지라고 한다.
  - 그래서 파이썬에서는 **라이브러리를 패키지라고 한다.** (재사용가능한 모듈들을 모아 놓은 것이 패키지이므로)
- 물리적으로는 모듈 파일들을 모아놓은 디렉토리(폴더)가 패키지이다.
- python 3.3 이전 버전은 package 디렉토리에 **__init__.py** 파일을 그 디렉토리에 반드시 위치시켜야 한다.
  - 3.3 이후에는 위치시킬 필요는 없지만 package안의 모듈들의 import 관련 설정을 해야 하는 경우에는 `__init__.py` 에 작성하고 위치시킨다.
- **Root Package**
  - 전체 모듈들을 담고 있는 최상위 패키지(디렉토리)
  - 패키지 내의 속한 패키지를 통칭 **sub package** 라고 한다.
  - Root package를 제외한 모든 package들은 다 sub package가 된다.

import

함수, 클래스 정의란

1. 함수, 클래스를 구현한다.
2. 구현된 함수, 클래스를 파이썬 실행환경에 등록한다.
   - 등록하는 것은 메모리에 올리는(loading) 작업이다.
   - 메모리에 올리기 위해서는 실행시켜서 파이썬 실행환경이 읽도록 해야 한다.

- 파이썬 실행환경에 등록된 함수와 클래스만 호출해서 사용할 수 있다.

import 란

- 파이썬 모듈 파일에 정의된 변수, 함수, 클래스들을 사용하기 위해 **파이썬 실행환경에 등록하는 작업**을 말한다.
- 현재 프로그램 모듈의 것들이 아니라 **다른 모듈에 있는 것들은 사용하기 위해 import 작업을 먼저 해야 한다.**
- 모듈을 import 하면 모듈의 내용이 실행되면서 그 안에 구현된 변수, 함수, 클래스들이 파이썬 실행환경에 그 모듈 이름을 namespace로 하여 등록된다.

> - **namespace**
>   - 여러개의 객체(존재하는 무언가)를 하나로 묶어 주면서 구분자 역할을 하는 이름을 주는 것을 namespace라고 한다.
>   - namespace를 이용해 각 그룹들의 객체들을 구분할 수 있다. 그래서 같은 이름의 객체들을 사용할 수 있다.
>   - 파이썬에서는 모듈에 정의된 변수, 함수, 클래스 들을 실행환경에 등록할 때 모듈명을 namespace로 묶어서 등록한다.
>   - 위키백과 참고 (https://ko.wikipedia.org/wiki/%EC%9D%B4%EB%A6%84%EA%B3%B5%EA%B0%84)

import 구문

- 기본구문
  - `[from 사용할 것의 경로] import 사용할 것 [as 별칭] [, 사용할 것..]`
  - []: 생략 가능한 구문
  - 사용할 것
    - 모듈
    - 모듈안에 정의된 변수, 함수, 클래스

1. 모듈 import

```
import 모듈   # 하나의 모듈 import.
import 모듈 as 별칭 # namespace의 이름을 모듈명이 아니라 별칭으로 지정한다.
import 모듈_1, 모듈_2 # 여러개 모듈 import.','를 구분자로 나열한다.
```

- 모듈을 import 하고 그 안에 함수, 클래스들을 사용할 때는 모듈명이 namespace 역할을 하기 때문에 `모듈명.함수()`, `모듈명.Class` 구문으로 호출한다.
- 별칭(Alias)를 주면 namespace로 별칭을 사용한다.
- **예**

```
import test_module
import my_module as mm
# test_module의 hello() 함수 호출시
test_module.hello()
# my_module은 mm 별칭을 지정했으므로 mm을 namespace로 사용한다.
p = mm.Person('홍길동', 30) # my_module의 Person 클래스 객체 생성
```

In [ ]:

```
```

2. 모듈내의 특정 항목만 import

```
from 모듈 import 함수  # 함수/클래스가 있는 모듈과 함수를 분리해서 import한다.
from 모듈 import 클래스
from 모듈 import 함수_1, 함수_2, 클래스
from 모듈 import *
```

- 모듈에 정의된 **일부 함수나 클래스만 사용할 경우** 개별적으로 import 할 수있다.
- `from 모듈 import 함수` 구문으로 import 하면 import한 **함수나 클래스들이 현재 실행중인 모듈의 namespace로 들어간다. 그래서 모듈명없이 바로 호출 할 수 있다.**
- `*`를 이용하면 그 모듈의 모든 함수/클래스들을 현재 실행중인 namespace에 추가해 사용할 수 있게 해준다. 이 방식은 **이름 충돌의 가능성이 있기때문에 추천되지 않는다.**

In [ ]:

```
```

3. 패키지에 속한 모듈 import

```
import 패키지명.모듈
from 패키지명 import 모듈
from 패키지명 import 모듈_1, 모듈_2
from 패키지명.모듈 import 함수
from 패키지명.모듈 import 클래스
from 패키지명.모듈 import 함수_1, 함수_2, 클래스
from Root패키지.Sub패키지1.Sub패키지2 import 모듈        # 패키지가 계층구조로 되있을
경우 `.` 으로 이용해 나열한다.
from Root패키지.Sub패키지1.Sub패키지2.모듈 import 함수
from Root패키지.Sub패키지1.Sub패키지2.모듈 import 클래스
```

- 패키지에 속한 모듈을 import 할 때는 **from 절에 패키지를 import 절에 모듈을** 설정한다.
- **import 가능한 것은 변수, 모듈, 함수, 클래스 들이다. 패키지는 import 할수 없다.**

In [ ]:

```
```

import 된 모듈 찾는 경로 및 PYTHONPATH

- `import 모듈` 구문을 사용하면 파이썬 실행 환경은 모듈을 다음 경로에서 찾는다.

  1. 현재 실행중인 모듈(import 구문을 사용한 모듈)이 있는 경로
  2. 파이썬 실행환경에 등록된 경로
- 모듈을 찾는 순서는 다음에서 확인할 수 있다.

```
import sys      # 표준모듈 sys
print(sys.path) # 모듈을 찾는 경로를 저장한 list
```

- 위의 경로 이외에 파이썬 모듈이 있을 경우 PYTHONPATH 환경변수에 그 디렉토리 경로를 등록한다.
  1. sys.path 에 추가한다. (사용할 때 마다 추가해야 한다.)
  2. 운영체제 환경변수에 등록한다. (한번만 하면된다.)

In [ ]:

```
```

메인 모듈(Main Module)과 하위 모듈(Sub Module)

- **메인 모듈**
  - 현재 실행하고 있는 모듈
    - `python 모듈.py` 로 실행된 모듈을 말한다.
  - application 의 main logic을 처리한다.
- **하위 모듈 (Sub module)**
  - 메인 모듈에서 import 되어 실행되는 모듈
  - 모듈을 import하면 그 모듈을 실행 시킨다. 이때 모듈에 있는 실행코드들도 같이 실행된다. 이것을 방지 하기 위해 모듈이 메인 모듈로 실행되는지 하위 모듈로 실행되는지 확인이 필요하다.
- `__name__` 내장 전역변수
  - 실행 중인 모듈명을 저장하는 내장 전역변수
  - **메인 모듈은 'main'** 을 **하위 모듈은 모듈명(파일명)** 이 저장된다.
  - 모듈이 메인 모듈로 시작하는지 여부 확인 할 때 사용한다.

```
if __name__ == '__main__':
    # 메인모듈일 때만 실행할 코드 블록
```

In [ ]:

```
```