Python 모듈&패키지

키워드 : Python 모듈 (https://docs.python.org/ko/3.7/tutorial/modules.html),
Python 패키지 (https://docs.python.org/ko/3.7/tutorial/modules.html#packages)

우리는 파이썬 코드를 작성하다 보면, 비슷한(혹은, 완전히 같은) 기능을 하는 코드가 중복되게 발생하는 코드가 있을 수 있습니다. 필요할 때마다 코드를 `Ctrl + c, v`하는 것도 하나의 방법이지만, 만약 코드가 몇 백, 몇 천 줄일 경우엔 꽤나 생각하기 싫은 작업이 되겠죠..?^^ 그렇기 때문에, 원활한 유지 보수를 위해 코드를 하나의 파이썬 파일에 작성하는 것이 아닌, 여러 개의 파이썬 파일로 나눠서 작성하게 됩니다. 이때, 각각의 파이썬 파일들을 모듈화 혹은, 스크립트로 만든다는 표현을 하게 됩니다.

Python 모듈(Module)

모듈 안에는 함수, 클래스, 혹은 변수들이 정의될 수 있으며, 실행 코드를 포함할 수도 있습니다. 파이썬 코드를 논리적인 기능들로 나눠서 분리했다면, 작성된 모듈을 불러올 수 있는 코드를 작성 해야합니다.

모듈을 불러오는 기본 구조는 강의에서 살펴본 것 처럼, `import 모듈명` 형태로 불러올 수 있었습니다. 또한, `from 모듈명 import 함수명` 형태로 작성하면 해당 모듈안에 작성된 특정 함수만을 가져와 사용할 수 있었습니다.

아래 코드는, 파이썬에서 기본적으로 제공하는, `time`이라는 모듈 안에 들어있는 `strftime` 함수를 `import`하는 예시입니다.

```
from time import strftime as strf

print(strf('%Y-%m-%d')) # e.g., 2018-08-31
```

우선, `strftime` 함수는 날짜 정보를 원하는 문자열 포맷에 맞게 작성할 수 있도록 만들어주는 함수입니다.

위 코드 예시를 살펴보겠습니다. 모듈명만을 이용하여 `import` 하면 `모듈명.함수` 형식으로 함수를 호출해야 합니다. 하지만, `from 모듈명 import 함수, 함수,...` 형식으로 함수를 직접 `import` 하면, 모듈명을 작성하지 않고, 곧바로 함수 이름으로 호출할 수 있습니다. 또한, 함수나 모듈의 이름이 너무 길다면, 위 예시처럼 `as` 키워드를 이용해 약어로 사용할 수도 있습니다.

Python 패키지(Package)

패키지는 여러 개의 모듈(파이썬 파일)을 하나로 묶어서(폴더) 관리하는 방법입니다. 마찬가지로, `from 최상위패키지.서브패키지1,... import 모듈` 형태로 작성할 수 있습니다.

우선, 아래와 같은 폴더 구조로 파이썬 코드를 분리했다고 가정하도록 하겠습니다.

```
code/
    main.py
    sub.py
    package/
            pck.py
```

```
from code import main # code 폴더 밑의 main 모듈 import
from code import sub # code 폴더 밑의 sub 모듈 import
# code 폴더 밑의 서브 패키지인 package 폴더 안에 있는 pck 모듈 import
from code.package import pck
```

모듈에서 from과 import 구문에 대한 이해가 있으시다면, 위 코드는 어렵지 않게 이해할 수 있을 거라 생각합니다. 단, 주의할 점은, 패키지 안에 있는 모듈을 불러오는 방법과, 모듈 안에 함수를 불러오는 키워드가 같다는 점에 대해 주의해야 합니다. 또한, 패키지를 점(.)으로 나열한 경우(e.g., code.sub1.sub2) 가장 마지막에 작성한 항목을 제외한 모든 항목은 패키지이어야(code.sub) 합니다. 단, 마지막 항목의(sub2) 경우는, 패키지이거나, 모듈일 수도 있습니다.

파이썬에서 모듈을 import 하면 크게 세 가지 경로 순서대로 해당 모듈을 찾는다고 말씀드렸습니다.

1. 현재 디렉토리
2. 환경변수 PYTHONPATH에 지정된 경로
3. Python이 설치된 경로 및 그 밑의 라이브러리 경로

위의 순서는 우리가 직접 모듈을 구성하고 사용할 경우이거나, 파이썬 가상 환경을 따로 구성하지 않고 사용할 경우에 고려해야 하는 경우입니다. 여러 모듈을 하나의 패키지로 구성하기 위해서(패키징), 패키지에서 자주 사용되는 모듈은 같은 디렉토리에 저장합니다.

우리는 파이썬 소개 참고 자료에서 이야기 했듯이, 가상 환경을 구성하여 각종 패키지를 관리합니다. 또한, VS Code의 인터프리터 모드를 가상 환경을 타겟으로 작동하게 하기 때문에, 가상 환경이 실행되지 않은 상태에서 코드를 실행하면, importError라는 에러가 발생합니다.

참고로, pip로 다운로드 받은 패키지는 .venv -> lib -> site-packages 경로에서 확인할 수 있습니다.

스크립트 단독 실행

위의 내용을 살펴보면, 모듈은 각각의 기능은 나눠서 코드를 관리하는 방법임을 알 수 있습니다. 이렇게 코드를 나눠서 관리하고, 모듈을 사용하는 파이썬 파일에서는 해당 모듈의 함수나 클래스 등을 불러와서 사용할 수 있습니다. 때로는, 모듈로 사용되는 파이썬 파일 코드 자체에서도 단독으로 스크립트를 실행해서 어떤 동작을 진행하고 싶을 경우가 있습니다. (python3 module.py)

이럴 경우 사용할 수 있는 방법이 __name__이라는 키워드입니다. 파이썬의 __name__ 키워드는 파이썬이 내부적으로 사용하는 특별한 변수명입니다. 아래 코드의 예처럼 특정 파이썬 파일이 단독으로 컴파일되어 실행될 때 실행할 코드와, 파이썬 파일을 모듈로 import 하여 사용할 때를 구분 지어 사용할 수 있습니다. 단톡 스크립트 실행시에만 작동할 코드를 if __name__ == "__main__:" 구문 아래에 작성하면 됩니다.

```
# 여러 함수 로직 등 구성.
if __name__ == "__main__":
  # 단독 스크립트 실행시 작동될 코드 작성.
```

우리가 빠른 개발을 위해 사용할 수 있는 모듈은 굉장히 많습니다. 파이썬 언어의 표준 모듈(e.g., sys, time, dir, etc.) 뿐만 아니라, 우리가 이미 pip를 이용해 다운로드한 것처럼, 다양한 3rd party 패키지가 있습니다. 원하는 기능을 개발하기 위해서, 이미 많은 개발자들을 통해 검증되고, 개발된 다양한 패키지를 어떻게 활용하고 사용할 수 있을지 알아보고, 공부해보면 많은 도움이 될 거라 생각합니다! :)