모듈 설계, 설치와 활용

파이썬 프로그래밍

2020년 2학기

서경대학교 김진헌

01. 모듈의 설계

모듈 선언: _md.py

1

- 파이썬 모듈은 함수들의 선언을 나열해 놓은 파일만으로도 하나의 모듈이 완성된다.
- This routine plays two roles.
  1) module that can be imported in the other program to support add/sub functions
  2) standalone program that can be executed independently

```
function based add/sub, x=1, y=2: x+y=3, x-y=-1
```

⇐ _md.py를 메인 프로그램으로 수행하였을 경우 출력 메시지

모듈 '_md ' 의 소스 파일:_md.py

```python
def add(x, y): return x + y
def sub(x, y): return x - y

def main():
    x = 1; y = 2
    print(f"function based add/sub, x={x}, y={y}: x+y={add(x, y)}, x-y={sub(x, y)}")

if __name__ == "__main__": # main 프로그램의 자격인지 확인한다. 맞으면 True를 반납한다.
    main()                 # 메인 프로그램의 자격으로 수행될 때 호출된다.
else:                      # import 되는 순간 수행된다..
    import os
    _dir = os.getcwd()      # get Current Working Directory
    print("'_md.py' is being loaded as a module from the following directory:\n" + _dir)
```

02. 모듈 함수의 호출

메인 프로그램: md_test.py

- 이 모듈을 호출하는 루틴에서는 모듈의 이름을 지정해 import 하여 사용한다.

```
import os
import _md as m
print(f"\n1) module location: \n{m.__file__}")
print(f"\n2) current directory: \n{os.getcwd()}")
x = 1; y = 2
print(f"\n3) imported module based functions: \
x+y={m.add(x, y)}, x-y={m.sub(x, y)}, where x={x}, y={y}")
```

모듈 '_md'의 함수 add/sub를 이용하여 연산을 수행하는 메인 md_test.py

(import할 때 모듈 _md.py에서 출력한 결과)

```
'_md.py' is being loaded as a module from the following directory:
D:\Work\@@Python\LectureMaterials\04_(ch4)_조건문과 반복문\Ch04\module_test

1) module location:
D:\Work\@@Python\LectureMaterials\04_(ch4)_조건문과 반복문\Ch04\module_test\_md.py

2) current directory:
D:\Work\@@Python\LectureMaterials\04_(ch4)_조건문과 반복문\Ch04\module_test

3) imported module based functions: x+y=3, x-y=-1, where x=1, y=2
```

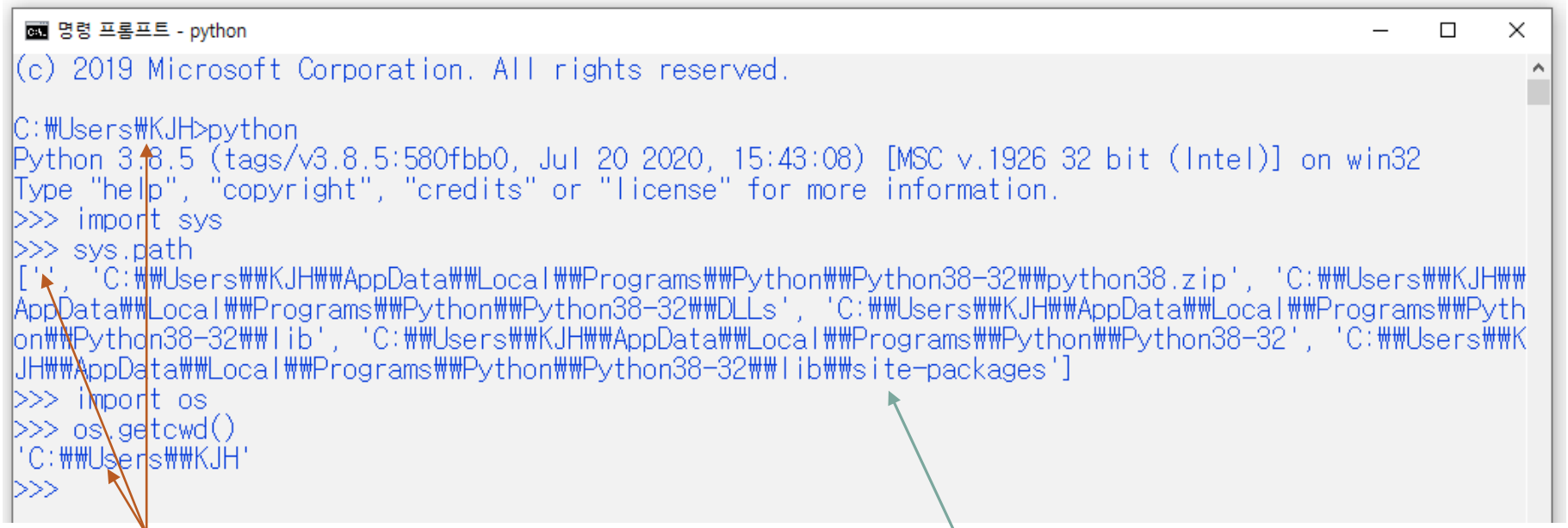
메인 프로그램 md_test.py의 print() 결과

03. 모듈의 위치

참조링트: 파이썬 모듈의 위치

- 모듈은 import 해서 사용하기 전에 미리 몇가지 규칙에 따라 호출 가능한 위치에 저장해 두어야 올바르게 load 할 수 있다.
- 메인 프로그램에서 모듈을 import 하면 다음과 같은 경로를 순서대로 검색한다.
  - 1. 현재 디렉토리
  - 2. 환경변수 PYTHONPATH에 지정된 경로
    - 개인용 모듈을 지정하여 사용할 때 외에는 사용할 필요 없음.
  - 3. PIP 설치 모듈 명령어가 사용하는 라이브러리 경로
    - 사례: D:₩Work₩W_PYTHON₩Python₩Python_3.7.6₩Lib₩site-packages
- 이러한 경로들은 모두 취합되어 시스템 경로인 sys.path에 리스트 형태로 저장된다. 따라서, 모듈이 검색되는 검색 경로는 sys.path를 체크하면 쉽게 알 수 있다.
- 모듈을 import 하면(좀더 자세히);
  - 1) sys.modules 에 등록되어 있는지 확인한다. 등록되어 있으면 로드한다. *11장에서 학습
  - 2) 1)에서 존재하지 않으면 sys.path의 디렉토리를 검색하면서 mylib 모듈을 찾는다.
    - sys.path에 있는 경로 순서대로 모듈을 찾아 import하다가 만약 끝까지 찾지 못하면 에러가 발생된다.

04. 모듈위치 확인하기(명령창)



* 현재 환경변수 PYTHONPATH는 설정에 의한 경로는 지정되어 있지 않다.

04. 모듈위치 확인하기(파이참)

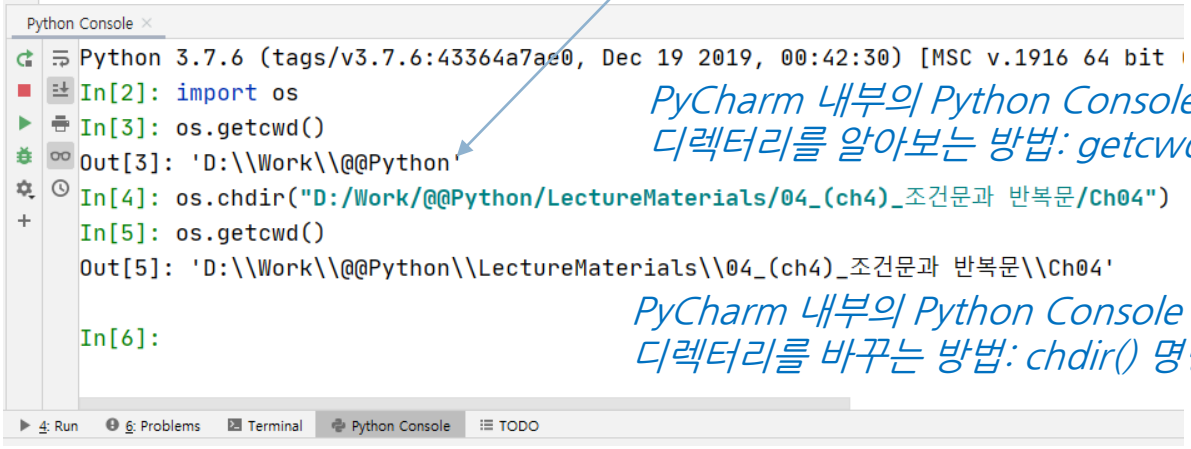
5

```
In[8]: import sys
In[9]: sys.path
Out[9]:
['C:\\Program Files\\JetBrains\\PyCharm Community Edition 2020.1.1\\plugins\\python-ce\\helpers\\pydev',
 'C:\\Program Files\\JetBrains\\PyCharm Community Edition 2020.1.1\\plugins\\python-ce\\helpers\\third_party\\thriftpy',
 'C:\\Program Files\\JetBrains\\PyCharm Community Edition 2020.1.1\\plugins\\python-ce\\helpers\\pydev',
 'D:\\Work\\W_PYTHON\\Python\\Python_3.7.6\\python37.zip',
 'D:\\Work\\W_PYTHON\\Python\\Python_3.7.6\\DLLs',
 'D:\\Work\\W_PYTHON\\Python\\Python_3.7.6\\lib',
 'D:\\Work\\W_PYTHON\\Python\\Python_3.7.6',
 'C:\\Users\\KJH\\AppData\\Roaming\\Python\\Python37\\site-packages',
 'D:\\Work\\W_PYTHON\\Python\\Python_3.7.6\\lib\\site-packages',
 'D:\\Work\\W_PYTHON\\Python\\Python_3.7.6\\lib\\site-packages\\pip-19.3.1-py3.7.egg',
 'D:\\Work\\W_PYTHON\\Python\\Python_3.7.6\\lib\\site-packages\\win32',
 'D:\\Work\\W_PYTHON\\Python\\Python_3.7.6\\lib\\site-packages\\win32\\lib',
 'D:\\Work\\W_PYTHON\\Python\\Python_3.7.6\\lib\\site-packages\\Pythonwin',
 'D:\\Work\\W_PYTHON\\Python\\Python_3.7.6\\lib\\site-packages\\IPython\\extensions',
 'D:\\Work\\@@Python',
 'D:/Work/@@Python']
```

설치된 모듈을 검색하는 순서…
sys.path를 보면 알 수 있다.

1. pip 명령으로 다운로드하는 모듈이 설치되는 곳

2. PyCharm 프로젝트에서 지정한 곳



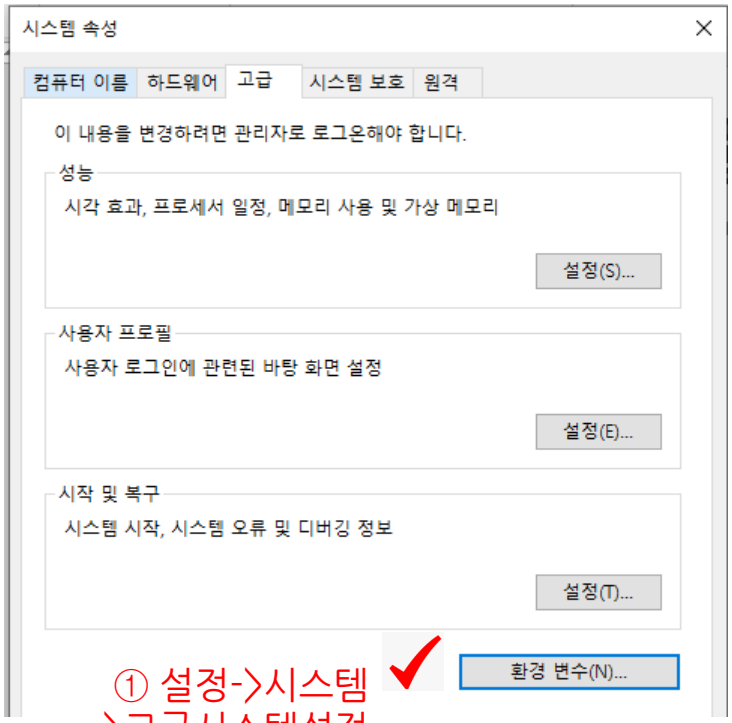
PyCharm 내부의 Python Console 창에서 현재 디렉터리를 알아보는 방법: getcwd() 명령

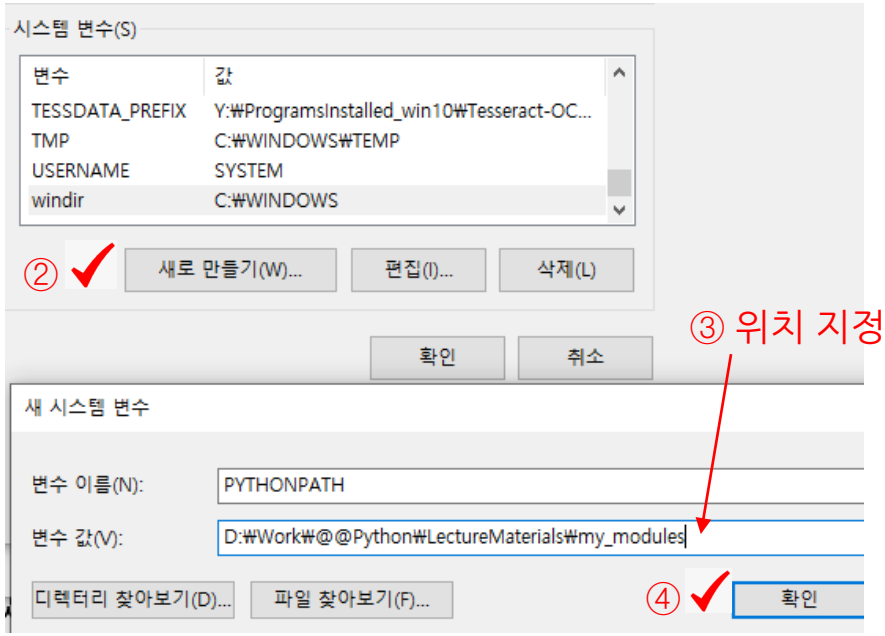
PyCharm 내부의 Python Console 창에서 현재 디렉터리를 바꾸는 방법: chdir() 명령

04. 모듈위치 확인하기(파이참)

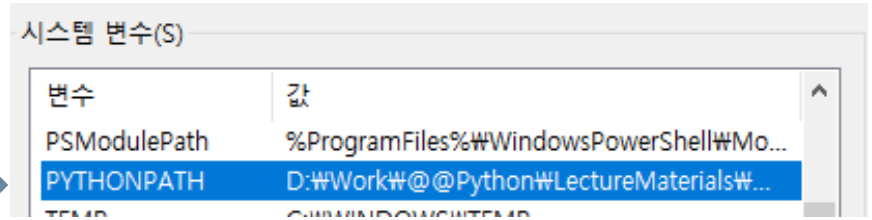
- 주의!!!
- PyCharm에서 어떤 소스 프로그램을 수행하면 CWD(Current Working Directory)는 소스프로그램이 위치한 곳이 된다.
- PyCharm에서의 python console 창은 프로젝트에서 정의한 폴더를 default 디렉터리로 사용한다.

05. 환경변수 PYTHONPATH 지정 방법





⑤확인



⑥확인: set 명령을 수행하면 아래와 같이 설정이 출력되는 것을 확인할 수 있다.

```
PyCharm Community Edition=C:₩Program Files₩JetBrains₩PyCha
PYTHONPATH=D:₩Work₩@@Python₩LectureMaterials₩my_modules
SESSIONNAME=Console
```

참고: PYD

- Python pyd 만들기
- import mylib
- # 이 코드를 수행할때 파이썬은 모듈을 로드하기 위해 다음의 절차에 따라 검색한다.
  - 1) sys.modules 에 등록되어 있는지 확인한다. 등록되어 있으면 로드한다.
  - 2) 1)에서 존재하지 않으면 sys.path의 디렉토리를 검색하면서 mylib 모듈을 찾는다.
- 다시 한번 요약하면, 외부 모듈을 import 하게되면 파이썬은
  - 1) 모듈이름과 일치하는 모듈을 찾는다.
  - 2) 모듈을 초기화 한다.
  - 3) 모듈이름을 지역이름 공간에 할당한다.