모듈과 패키지

안 화 수

1.모듈

- 모듈
  - 모듈이란 함수나 변수 또는 클래스 들을 모아 놓은 파일이다
  - 일반적으로는 "독자적인 기능을 갖는 구성 요소"를 의미
  - 서로 연관된 작업을 하는 코드들의 모임으로 작성 중인 모듈의 크기가 어느 정도 커지게 되면 일반적으로 관리 가능한 작은 단위로 다시 분할
  - 파이썬에서는 개별 소스 파일을 일컫는 말
  - 파이썬 프로그램으로 작성된 파일도 가능하고 C나 Fortran 등으로 만든 파이썬 확장파일도 모듈이 될 수 있음

- 모듈의 종류
  - 표준 모듈 : 파이썬 언어 패키지 안에 기본적으로 포함된 모듈 예) math, string
  - 사용자 생성 모듈 : 프로그래머가 직접 작성한 모듈
  - 서드 파티(3rd Party) 모듈 : 파이썬 재단도 프로그래머도 아닌 다른 프로그래머, 또는 업체에서 제공한 모듈
- 외부 모듈을 사용할 수 있도록 추가하는 방법은 import 파일명
- 확장자는 생략하고 파일에 있는 변수나 메소드는 파일명.변수 또는 파일명.함수()로 호출

1.모듈

- 자격(Qualified) 변수: 앞의 예에서 처럼 소속을 정확히 밝혀서 사용하는 변수
- 무자격(Unqualified) 변수: 소속을 밝히지 않고 사용하는 변수
- 파이썬은 모듈 가져오기를 수행하면 특별히 지정한 폴더에서 찾아가기 시작합니다.
- sys.path 변수에 그 순서가 기재되어 있습니다.

```
import sys
print(sys.path)     #파이썬 모듈들이 저장되어 있는 위치를 구해줌
```

결과

```
['C:□□Users□□Administrator□□python□□test',
    'C:□□Users□□Administrator□□python□□test',
    'C:□□Users□□Administrator□□AppData□□Local□□Programs□□Python□□Python35-
    32□□DLLs',
    'C:□□Users□□Administrator□□AppData□□Local□□Programs□□Python□□Python35-
    32□□lib',
    'C:□□Users□□Administrator□□AppData□□Local□□Programs□□Python□□Python35-
    32',
    'C:□□Users□□Administrator□□AppData□□Local□□Programs□□Python□□Python35-
    32□□lib□□site-packages',
    'C:□□Users□□Administrator□□AppData□□Local□□Programs□□Python□□Python35-
    32□□python35.zip']
```

1.모듈

- 직접 검색할 위치를 추가: sys.path.append("검색할 경로")
- 윈도우의 환경 변수에 추가
  PYTHONPATH = 검색경로;

- 리눅스의 환경 변수에 추가
  C Shell 인 경우: setenv PYTHONPATH 검색경로
  bash Shell 인 경우: export PYTHONPATH =검색경로

- 절대 경로를 이용한 가져오기
  - import math : math 모듈처럼 모듈의 이름만 가져온 경우로 math.해서 사용
  - from math import sin, cos, pi: math 모듈에서 sin, cos, pi 만 가져온 경우로 math.을 생략하고 사용 가능
  - from math import * : math 모듈의 모든 이름을 현재 위치로 가져옵니다.
  - import math as ma: math라는 모듈 이름 대신에 ma 사용
  - from math import pi as py: pi 대신에 py 사용
  - 하나의 모듈에서 여러 개의 이름을 가져올 때는 괄호 가능
  - 모듈 이름이 문자열로 되어 있는 경우 __import__(모듈이름) 으로 가져오는 것이 가능
  - 모듈은 한 번 가져오면 메모리에 적재된 상태가 되므로 다른 모든 모듈에서 사용이 가능

2.내장함수(Builtin Function)

- 내장 함수
  - 파이썬의 표준모듈에 내장되어 있는 함수들이다.
  - 내장함수는 import 구문을 사용하지 않고 사용할 수 있다.

- 내장 함수 확인
  https://docs.python.org/3.6/library/functions.html

  - input
  - print
  - type
  - int
  - str
  - range
  - max
  - min

3.표준내장모듈

- 표준 내장 모듈 : math모듈

- 내장 모듈 import 방법

1. import 모듈명
   import math
   math 모듈을 import하면 math모듈 안에 있는 모든 변수와 함수들을 사용할 수 있다.
   ex. math.pi
       math.factorial(3)

2. from 모듈명 import 변수 또는 함수명
   from math import factorial
   math 모듈안에 있는 factorial 함수만 import 한다는 의미를 가짐
   math 모듈안에 있는 다른 함수들은 사용 할 수 없다.
   ex. factorial(3)

3.표준내장모듈

```
import math
# from math import factorial


# 원주율(pi) 구하기
print(math.pi)

# 2의 3승 구하기
print('2의 3승=', math.pow(2, 3))

# 팩토리얼(factorial) 구하기
print('3!=',math.factorial(3))
print(math.factorial(984))

# ceil()함수 : 올림기능
print(math.ceil(3.1))

# floor(): 내림기능
print(math.floor(3.9))

# sqrt() : 제곱근
print(math.sqrt(5))
```

3.표준내장모듈

- 표준 내장 모듈
  - 파이썬의 표준모듈에 내장되어 있는 함수들이다.
  - 표준모듈에 내장된 함수는 import 구문을 사용해서 사용할 수 있다.

- 내장 모듈 종류
  - sys
  - pickle
  - os
  - shutil
  - glob
  - tempfile
  - time
  - calendar
  - random
  - webbrowser

3.표준내장모듈

- sys 모듈

```
import sys

print(sys.argv)

# sys.path는 파이썬 모듈들이 저장되어 있는 위치를 구해줌
print(sys.path)

# 강제로 스크립트 종료하기
sys.exit()
```

3.표준내장모듈

- calendar 모듈

```
import calendar

# calendar()함수는 해당 연도의 달력을 리턴함
cal = calendar.calendar(2019)
print(cal)

# prcal() 함수는 2019년 달력을 출력해줌
calendar.prcal(2019)

# prmonth() : 특정 연도의 특정 월에 대한 달력을 출력
calendar.prmonth(2019,2)

# weekday() : 날짜에 해당하는 요일 정보를 리턴
# 월(0),화(1),수(2),목(3),금(4),토(5),일(6)
weekday = calendar.weekday(2019,2,28)
print('weekday:', weekday)
```

3.표준내장모듈

- random 모듈

```
# 난수 발생
import random

# 0.0 ~ 1.0 사이의 난수를 발생
r1=random.random()
print('r1:',r1)

# 1 ~ 10 사이의 난수를 발생
r2=random.randint(1,10)
print('r2:', r2)

# 1 ~ 45 사이의 난수를 발생
r3=random.randint(1,45)
print('r3:',r3)

# list의 항목중 1개를 추출
list = ['빨강','주황','노랑','초록','파랑','남색','보라']
r4=random.choice(list)
print('r4:',r4)
```

3.표준내장모듈

❖ time 모듈

```
import time

#time()은 UTC(Universal Time Coordinated 협정 세계 표준시)를
# 이용하여 현재 시간을 실수 형태로 리턴하는 함수.
# 1970년 1월 1일 0시 0분 0초를 기준으로 지난 시간을 초 단위로 리턴.
print(time.time())

#localtime()은 time()에 의해서 반환된 실수값을 이용해서
# 연도, 월, 일, 시, 분, 초,.. 의 형태로 바꾸어 주는 함수.
print(time.localtime(time.time()))

#asctime()은 localtime()에 의해서 반환된 튜플 형태의 값을
# 인수로 받아서 날짜와 시간을 알아보기 쉬운 형태로 리턴하는 함수.
print(time.asctime(time.localtime(time.time())))

# ctime()은 현재 시간을 간단하게 리턴하는 함수.
print(time.ctime())
```

3.표준내장모듈

- time 모듈

```
#strftime('출력할 형식 포맷 코드', time.localtime(time.time()))
print(time.strftime('%x', time.localtime(time.time())))
print(time.strftime('%c', time.localtime(time.time())))
print(time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time())))

#sleep() 함수를 사용하면 일정한 시간 간격을 두고 실행
for i in range(10):
    print(i)
    time.sleep(1)        # 1초 간격으로 실행
```

3.표준내장모듈

- webbrowser 모듈

```
import webbrowser

#webbrowser는 자신의 시스템에서 사용하는 기본 웹 브라우저가
# 자동으로 실행되게 하는 모듈이다
webbrowser.open("http://google.com")


#open_new 함수는 이미 웹 브라우저가 실행된 상태이더라도
# 새로운 창으로 해당 주소가 열리도록 한다.
webbrowser.open_new("http://naver.com")
```

4.사용자 정의 모듈

❖ 사용자 정의 모듈 파일 생성 : 예1

➢ mymath.py # 모듈 파일 생성

```
mypi = 3.14

def area(r):
        return mypi * r * r
```

➢ test.py

```
import mymath

print(mymath.mypi)

print(mymath.area(5))
```

4.사용자 정의 모듈

- 사용자 정의 모듈 파일 생성 : 예2

- calculator.py # 모듈 파일 생성

```
def plus(a, b):
        return a + b

def minus(a, b):
        return a - b

def multiply(a, b):
        return a * b

def divide(a, b):
        return a / b
```

4.사용자 정의 모듈

❖ 모듈 불러오는 파일 생성

➢ tester.py # 모듈 파일 불러오는 파일 생성

```
#calculator.py 모듈파일 불러오기

 #import 모듈이름
import calculator

print(calculator.plus(10, 5))
print(calculator.minus(10, 5))
print(calculator.multiply(10, 5))
print(calculator.divide(10, 5))
```

4.사용자 정의 모듈

- 모듈 불러오는 파일 생성

- tester2.py # 모듈 파일 불러오는 파일 생성

```
#calculator.py 모듈파일 불러오기

#from 모듈이름  import 변수 또는 함수명
from calculator import plus
from calculator import minus

print(plus(10, 5))
print(minus(10, 5))
# print(multiply(10, 5))
# print(divide(10, 5))
```

4.사용자 정의 모듈

❖ 모듈 불러오는 파일 생성

➢ tester3.py # 모듈 파일 불러오는 파일 생성

```
#calculator.py 모듈파일 불러오기

#from 모듈이름  import 변수 또는 함수명
from calculator import plus, minus

print(plus(10, 5))
print(minus(10, 5))
#print(multiply(10, 5))
#print(divide(10, 5))
```

4.사용자 정의 모듈

- 모듈 불러오는 파일 생성

- tester4.py # 모듈 파일 불러오는 파일 생성

```
#calculator.py 모듈파일 불러오기
#와일드카드(*)를 이용해서 모듈안에 들어 있는 모든 변수와 함수를  import

#from 모듈이름  import  *
from calculator import *

print(plus(10, 5))
print(minus(10, 5))
print(multiply(10, 5))
print(divide(10, 5))
```

4.사용자 정의 모듈

❖ 모듈 불러오는 파일 생성

➢ tester5.py # 모듈 파일 불러오는 파일 생성

```
#calculator.py 모듈파일 불러오기

#from 모듈이름  as  새로운 모듈이름(별칭)
import calculator as c

print(c.plus(10, 5))
print(c.minus(10, 5))
print(c.multiply(10, 5))
print(c.divide(10, 5))
```

5.외부 모듈

- 외부모듈 설치

```
c:₩> pip install numpy

c:₩> pip install pandas

c:₩> pip install tensorflow
```

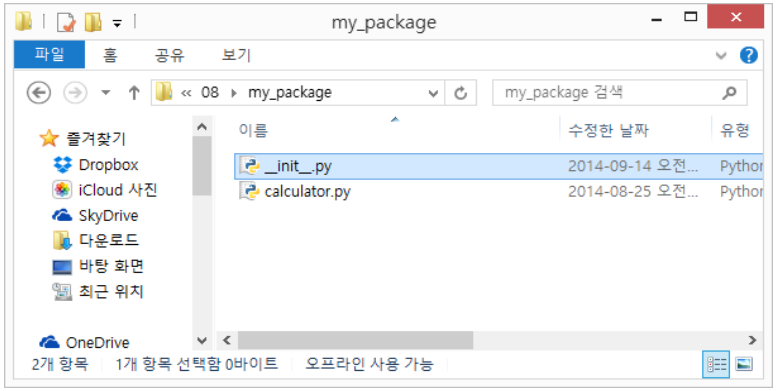
- 외부모듈 불러오는 형식

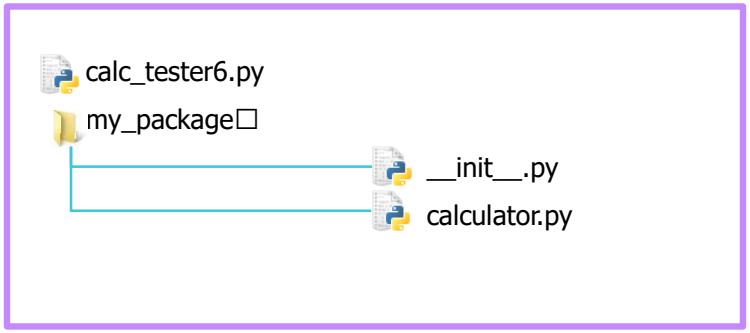
```
import numpy as np

import pandas as pd

import tensorflow as tf
```

6.패키지

- 패키지
  - 모듈을 모아놓는 디렉토리
  - 모듈 꾸러미로 해석하면 이해하기 편함
  - 디렉토리가 "파이썬의 패키지"로 인정받으려면 __init__.py 파일을 그 경로에 갖고 있어야 함
  - __init__.py 파일에는 패키지를 초기화하는 어떠한 파이썬 코드도 포함할 수 있습니다.
  - 패키지에서 특정한 모듈을 가져올 때는 아래 그림과 같은 경우

from my_package import calculator





6.패키지

- 프로젝트에 my_package를 생성
- 위에서 생성한 패키지 안에 calculator.py 파일을 생성하고 내용을 작성

```
def plus(x, y):
    return x+y

def minus(x, y):
    return x-y
```

- test.py 파일을 생성하고 위의 패키지를 사용하는 코드를 작성

```
from my_package import calculator
result = calculator.plus(10, 30)
print("결과:{0}".format(result))
result = calculator.minus(10, 30)
print("결과:{0}".format(result))
```

6.패키지

- 보통의 경우 init__.py 파일은 대개 비워둠

- 이 파일을 손대는 경우는 __all__이라는 변수를 조정할 때 정도
  - __all__은 다음과 같은 코드를 실행할 때 패키지로부터 반입할 모듈의 목록을 정의하기 위해 사용

__all__ =[모듈 나열]

from .import 모듈이름

- import *은 사용을 자제하는 것이 좋음.

from 패키지 import *

6.패키지

❖ 프로젝트에 my_package 내부의 __init__.py 파일을 수정

```
__all__ =['calculator']
from . import calculator
```

❖ test.py 파일을 수정

```
import my_package
result = my_package.calculator.plus(10, 30)
print("결과:{0}".format(result))
result = my_package.calculator.minus(10, 30)
print("결과:{0}".format(result))
```

6.패키지

- site-packages
  - 파이썬의 기본 라이브러리 패키지 외에 추가적인 패키지를 설치하는 디렉토리
  - 각종 서드 파티 모듈을 바로 이 곳에 설치함

- site-packages 확인

```
>>> import sys
>>> sys.path
['', 'C:₩₩Python34₩₩Lib₩₩idlelib',
'C:₩₩WINDOWS₩₩SYSTEM32₩₩python34.zip', 'C:₩₩Python34₩₩DLLs',
'C:₩₩Python34₩₩lib', 'C:₩₩Python34',
'C:₩₩Python34₩₩lib₩₩site-packages']
```

site-package는 파이썬이 기본적으로 모듈을 탐색하는 경로에 포함되어 있습니다.

6.패키지

- 패키지 안에 모듈 파일 불러오는 예제

1. my_package 패키지 생성
2. my_package 패키지 안에 calculator.py 모듈 파일 생성

my_package - calculator.py

6.패키지

- my_package 패키지 – calculator.py

- calculator.py # 모듈 파일 생성

```
def plus(a, b):
        return a + b

def minus(a, b):
        return a - b

def multiply(a, b):
        return a * b

def divide(a, b):
        return a / b
```

6.패키지

❖ 패키지 안에 모듈 파일 불러오는 방법

my_package 패키지 안에 calculator.py 모듈 파일 불러오는 파일 생성

➢ tester6.py

```
#my_package패키지 안에 있는 calculator.py 모듈파일 불러오기

# 방법1.
# from  패키지명  import  모듈파일명
from my_package import calculator
print(calculator.plus(10, 5))
print(calculator.minus(10, 5))
print(calculator.multiply(10, 5))
print(calculator.divide(10, 5))
print()
```

6.패키지

❖ 패키지 안에 모듈 파일 불러오는 방법

➢ tester6.py

```
# 방법2.
# import  패키지명.모듈명
import my_package.calculator
print(my_package.calculator.plus(10, 5))
print(my_package.calculator.minus(10, 5))
print(my_package.calculator.multiply(10, 5))
print(my_package.calculator.divide(10, 5))
print()

# 방법3.
# import  패키지명.모듈명  as  별칭명
import my_package.calculator  as  cal
print(cal.plus(10, 5))
print(cal.minus(10, 5))
print(cal.multiply(10, 5))
print(cal.divide(10, 5))
```

6.패키지

- 모듈 불러오는 형식

```
import matplotlib.pyplot as plt

plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```

- 모듈 설치

```
c:₩> pip install matplotlib
```

6.패키지

- matplotlib 모듈 실행 결과