

# Day 5

- Property 사용하기
  - 값을 가져오는 getter
  - 값을 저장하는 setter

```
class Person:
    def __init__(self):
        self.__age = 0

    def get_age(self):           #getter
        return self.__age

    def set_age(self, value):    #setter
        self.__age = value
```

```
In [8]: jeon = Person()
```

```
In [9]: jeon.set_age(35)
```

```
In [10]: print(jeon.get_age())
```

35

## ■ Property 사용하기

- @property : 값을 가져오는 메소드에 붙인다.
- @메소드이름.setter : 값을 저장하는 메소드에 붙인다.

```
class Person:
    def __init__(self):
        self.__age = 0

    @property
    def age(self):          #getter
        return self.__age

    @age.setter
    def age(self, value):  #setter
        self.__age = value
```

```
In [4]: jeon = Person()
```

```
In [5]: jeon.age = 35
```

```
In [6]: print(jeon.age)
```

35

## ■ 클래스 관계

- is – a 관계 : 상속 관계

- ✓ 명확하게 같은 종류, 동등한 관계일 때
- ✓ ‘학생은 사람이다.’라고 했을 때 말이 되면 동등한 관계
- ✓ Student **is a** Person

- has – a 관계 : 포함 관계

- ✓ 사람 목록을 관리하는 클래스 만든다면, 리스트 속성에 Person 객체를 넣어서 관리
- ✓ 같은 종류에 동등한 관계일 때는 상속, 그 이외에는 속성에 인스턴스를 포함

```
In [12]: class Person:
          pass

          class Student(Person):
              pass
```

## ■ 모듈과 패키지 사용하기

- 모듈(module)은 함수, 변수, 클래스를 담고 있는 파일
- 패키지(package) 여러 모듈을 묶은 것

## ■ import 로 모듈 가져오기

- 여러 개 모듈을 가져올 때는 콤마(,)로 구분

✓ import 모듈

✓ import 모듈1, 모듈2

✓ 모듈.변수

✓ 모듈.함수()

```
In [16]: import math
```

```
In [17]: math.pi
```

```
Out[17]: 3.141592653589793
```

```
In [18]: math.sqrt(4.0)
```

```
Out[18]: 2.0
```

- import as 로 모듈 이름 지정하기
  - 앞선 예제에서 math를 입력하고 싶지 않을 때
  - import 모듈 as 이름

```
In [20]: import math as m
```

```
In [21]: m.pi
```

```
Out[21]: 3.141592653589793
```

```
In [22]: m.sqrt(2)
```

```
Out[22]: 1.4142135623730951
```

## ■ from import 로 모듈 일부만 가져오기

- from 모듈 import 변수 (or 함수, or 클래스)

```
In [23]: from math import pi
```

```
In [24]: pi
```

```
Out[24]: 3.141592653589793
```

```
In [25]: from math import sqrt
```

```
In [26]: sqrt(2)
```

```
Out[26]: 1.4142135623730951
```

- math 모듈에서 가져올 변수와 함수가 여러 개 일 경우
  - ✓ import 뒤에 가져올 변수, 함수, 클래스를 콤마로 구분
  - ✓ from 모듈 import 변수, 함수, 클래스

```
In [27]: from math import pi, sqrt
```

```
In [28]: pi
```

```
Out[28]: 3.141592653589793
```

```
In [29]: sqrt(4)
```

```
Out[29]: 2.0
```

- from import 로 모듈 일부만 가져오기
  - 모든 변수, 함수, 클래스를 가져올 경우
  - from 모듈 import \*

```
In [30]: from math import *
```

```
In [31]: pi
```

```
Out[31]: 3.141592653589793
```

- 모듈 일부 가져오면서 이름 지정하기
  - from 모듈 import 변수 as 이름

```
In [32]: from math import sqrt as s
```

```
In [33]: s(4)
```

```
Out[33]: 2.0
```



- 모듈 일부 가져오면서 이름 지정하기
  - 여러 개 가져와 이름 지정할 경우
  - `from 모듈 import 변수 as 이름1, 함수 as 이름2, 클래스 as 이름3`

```
In [36]: from math import pi as p, sqrt as s
```

```
In [37]: p
```

```
Out[37]: 3.141592653589793
```

```
In [38]: s(2)
```

```
Out[38]: 1.4142135623730951
```

## ■ import 로 패키지 가져오기

- 패키지는 특정 기능과 관련된 여러 모듈을 묶은 것.
- 패키지 안에 들어있는 모듈도 import 를 사용하여 가져옴
  - ✓ import 패키지.모듈
  - ✓ import 패키지.모듈1, 패키지.모듈2
  - ✓ 패키지.모듈.변수
  - ✓ 패키지.모듈.함수

```
In [39]: import urllib.request
```

```
In [44]: response = urllib.request.urlopen('http://www.google.co.kr')  
         response.status
```

```
Out[44]: 200
```

- import as로 패키지 모듈 이름 지정하기
  - import 패키지.모듈 as 이름

```
In [45]: import urllib.request as r
```

```
In [46]: response = r.urlopen('http://www.google.co.kr')  
response.status
```

```
Out[46]: 200
```

- from import
  - from 패키지.모듈 import 변수
- 패키지 모듈에서 모든 변수, 함수, 클래스 가져오기
  - from 패키지.모듈 import \*

- 패키지 모듈의 일부를 가져와서 이름 지정
  - `from 패키지.모듈 import 변수 as 이름`
  - `from 패키지.모듈 import 변수 as 이름1, 함수 as 이름2, 클래스 as 이름3`

- 파이썬 패키지 인덱스에서 패키지 설치하기
  - PyPI (Python Package Index) 통해 다양한 패키지를 설치할 수 있다.
- pip 설치 하기
  - pip는 파이썬 패키지 인덱스의 패키지 관리 명령어
  - 윈도우용 파이썬에는 기본 내장
  - 사용방법
    - ✓ pip install 패키지
    - ✓ 윈도우키 + R , cmd입력

C:\> 명령 프롬프트

```
Microsoft Windows [Version 10.0.17763.615]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\Users\swedu>  
C:\Users\swedu>  
C:\Users\swedu>pip install requests
```

## ■ pip 설치 하기

- -m 옵션을 지정해서 pip 실행
- -m 옵션은 모듈을 실행하는 옵션이며 pip 도 모듈

```
C:\Users\swedu>python -m pip install requests
```

## ■ 모듈과 패키지 만들기

- 2의 거듭제곱 모듈 만들기
- Ai 폴더(c:\Ai) 안에 square2.py 파일로 저장

square2.py - C:/Ai/square2.py (3.7.3)

File Edit Format Run Options Window Help

```
base = 2
```

```
def square(n):  
    return base ** n
```

```
|
```

- Ai 폴더(c:\Ai) 안에 main.py 파일로 다음을 저장

main.py - C:/Ai/main.py (3.7.3)

File Edit Format Run Options Window Help

```
import square2
```

```
print(square2.base)
```

```
print(square2.square(3))
```

```
2
```

```
8
```

```
>>> |
```

```
main.py - C:/Ai/main.py (3.7.3)
File Edit Format Run Options Window Help
from square2 import base, square

print(base)
print(square(3))
|
```

## ■ 모듈에 클래스 작성하기

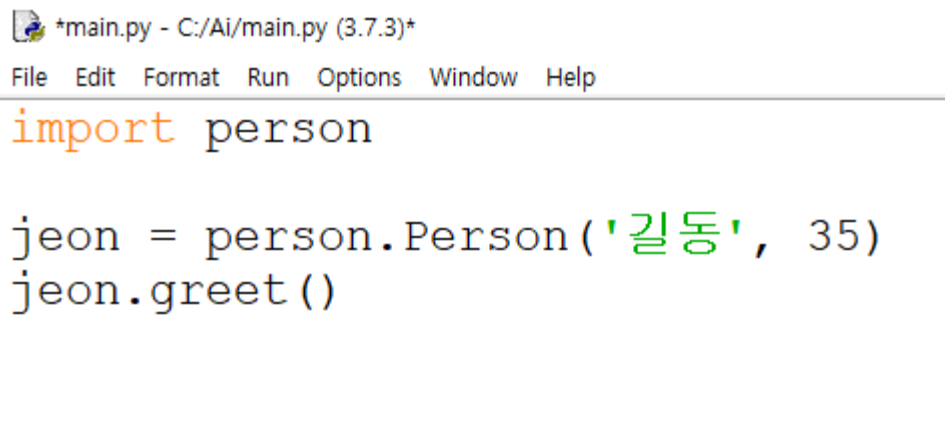
- Ai 폴더(c:\Ai) 안에 person.py 파일로 저장

```
person.py - C:\Ai\person.py (3.7.3)
File Edit Format Run Options Window Help
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print('안녕하세요.', self.name)
```



## ■ main 파일 수정

A screenshot of a Python IDE window titled '\*main.py - C:/Ai/main.py (3.7.3)\*'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor shows the following Python code:

```
import person

jeon = person.Person('길동', 35)
jeon.greet()
```

```
=====
```

```
안녕하세요. 길동
```

```
>>>
```

## ■ 모듈 시작점 알아보기

- 다음 코드를 작성하여 Ai 폴더에 저장(파일명 hello.py)

 \*hello.py - C:/Ai/hello.py (3.7.3)\*

File Edit Format Run Options Window Help

```
print('hello 모듈 시작')  
print('hello.py __name__:' __name__)  
  
print('hello 모듈 끝')
```

- 다음 코드를 작성하여 Ai 폴더에 저장(파일명 main.py)

 \*main.py - C:/Ai/main.py (3.7.3)\*

File Edit Format Run Options Window Help

```
import hello  
  
print('main.py __name__ : ', __name__)
```

- hello 모듈을 import 하면 hello 모듈의 내용이 실행
  - `__name__`에 모듈이 이름이 출력

```
===== RE
hello 모듈 시작
hello.py __name__: hello
hello 모듈 끝
main.py __name__ : __main__
>>>
```

- 파이썬 인터프리터가 최초로 실행한 스크립트 파일의 `__name__`에는 `__main__` 이 들어간다.(프로그램의 시작점 의미)

```
C:\#\Ai>python main.py
hello 모듈 시작
hello.py __name__: hello
hello 모듈 끝
main.py __name__ : __main__


C:\#\Ai>python phello.py
python: can't open file 'phello.py': [Errno 2] N

C:\#\Ai>python hello.py
hello 모듈 시작
hello.py __name__: __main__
hello 모듈 끝

C:\#\Ai>
```

## ■ 모듈과 시작점

- 최초 시작 스크립트 파일과 모듈의 차이가 없음
- 스크립트 파일이 시작점도 될 수 있고, 모듈도 될 수 있다.
- `__name__` 변수를 통해 현재 스크립트 파일이 시작점인지 모듈인지 판단
- `if __name__ == '__main__':`  
    `__name__` 변수 값이 `'__main__'` 인지 확인하는 코드는 현재 파일이  
    프로그램의 시작점이 맞는지 판단  
    파일이 메인 프로그램으로 사용될 때와 모듈로 사용될 때를 구분하기 위한  
    용도로 사용

 calc.py - C:/Ai/calc.py (3.7.3)

File Edit Format Run Options Window Help

```
def add_(x, y):  
    return x + y  
  
def mul_(x, y):  
    return x * y  
  
if __name__ == '__main__':  
    print(add_(1, 2))  
    print(mul_(1, 2))
```

```
3  
2  
>>> |
```

- calc.py를 모듈로 사용하면

```
>>> import calc
>>>
>>> |
```

- 아무것도 출력이 나오지 않는다.

\_\_name\_\_ 변수 값이 '\_\_main\_\_' 아니기 때문

- 파일을 모듈로 사용할 경우 calc.add 또는 calc.mul 함수만 사용하는 것이 목적이 되므로 1, 2의 합과 곱을 출력하는 코드는 필요 없음

```
>>> calc.add_(3, 4)
7
>>> calc.mul_(3, 4)
12
>>> |
```

## ■ 패키지 만들기

- 모듈은 파일이 한 개지만, 패키지는 폴더로 구성
- C:\Ai 폴더 밑에
- main.py 파일
- calpkg 폴더 (폴더 안에 \_\_init\_\_.py, operation.py, geometry.py) 로 구성



## ■ 패키지 만들기

- C:\Ai 폴더 안에 calcpkg 폴더 만든다.
- calcpkg 폴더 안에 \_\_init\_\_.py 파일 생성
- calcpkg 폴더 안에 operation.py 파일 생성

operation.py - C:/Ai/calcpkg/operation.py (3.7.3)

File Edit Format Run Options Window Help

```
def add_(x, y):  
    return x + y  
  
def mul_(x, y):  
    return x * y
```

geometry.py - C:/Ai/calcpkg/geometry.py (3.7.3)

File Edit Format Run Options Window Help

```
def t_area(x, y):  
    return x * y / 2  
  
def r_area(x, y):  
    return x * y
```

- calcpkg 폴더 안에 geometry.py 파일 생성

- C:\WAI 폴더 안에 main.py 생성

main.py - C:/Ai/main.py (3.7.3)

File Edit Format Run Options Window Help

```
import calcpkg.operation
import calcpkg.geometry
```

```
print(calcpkg.operation.add_(1, 2))
print(calcpkg.operation.mul_(1, 2))
```

```
print(calcpkg.geometry.t_area(3, 4))
print(calcpkg.geometry.r_area(3, 4))
```

```
3
2
6.0
12
>>>
```