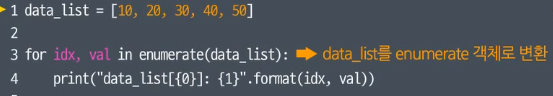
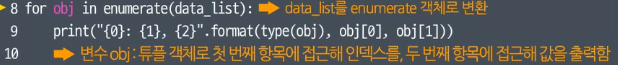
내장함수

* divmod()  
  첫 번째 인자를 두 번째 인자로 나눴을 때의 몫과 나머지를 **튜플** 객체로 반환하는 함수.
* all() (and 여러 개 버전인듯)  
  반복 가능한 자료형을 인자로 전달하여 항목 모두가 True이면 True, False로 평가되는 항목이 하나라도 있으면 False를 반환하는 함수.
* any() (or 여러 개 버전인듯)  
  반복 가능한 자료형을 인자로 전달하여 항목 모두가 False이면 False, True로 평가되는 항목이 하나라도 있으면 True를 반환하는 함수.
* enumerate()  
  List, Tuple, 문자열과 같은 시퀀스형을 입력받아 인덱스를 포함하는 튜플 객체를 항목으로 구성하는 enumerate 객체를 반환하는 함수.



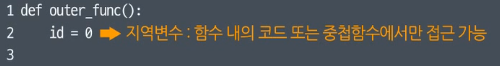


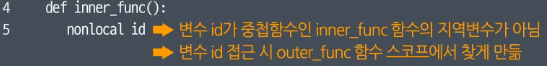


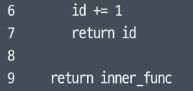
* filter()  
  조건에 해당하는 항목을 걸러내는 함수.  
  (ex> ret\_val = filter(iseven{Boolean}, numbers{iterable})
* sorted()  
  반복 가능한 자료형을 인자로 전달받아 항목들로부터 정렬된 **리스트**를 생성해 반환하는 함수.
* zip()  
  둘 이상의 반복 가능한 자료형을 인자로 전달받아, 동일 위치의 항목을 묶어 튜플을 항목으로 구성하는 zip 객체를 생성하는 함수.
* chr()  
  정수 형태의 유니코드 값을 인자로 받아 해당 코드의 문자를 반환하는 함수.
* ord()  
  문자를 인자로 전달 받아 유니코드 값(10진 정수)을 반환하는 함수.
* dir()  
  인자로 전달된 객체가 가지고 있는 변수, 메서드와 같은 속성 정보를 리스트 객체로 반환,  
  인자를 전달하지 않고 호출하면 현재 지역 스코프에 대한 정보를 리스트 객체로 반환.
* Isinstance()  
  첫 번째 인자로 전달된 객체가 두 번째 인자로 전달된 클래스의 인스턴스인지에 대한 여부를 True/False로 반환하는 함수.
* Issubclass()  
  첫 번째 인자로 전달된 클래스가 두 번째 인자로 전달된 클래스의 서브클래스인지에 대한 여부를 True/False로 반환하는 함수.
* eval()  
  실행 가능한 표현식의 **문자열**을 인자로 전달받아 해당 문자열의 표현식을 실행한 **결과값을 반환**하는 함수.

함수

* 언팩 연산자(\*)  
  매개변수의 개수를 가변적으로 사용할 수 있도록 함  
  매개변수를 적용 시 **튜플** 형식으로 처리 가능  
  \* 가변형 매개변수를 가장 마지막 매개변수로 지정해야 부작용 없이 사용할 수 있다.  
  (ex> def calc\_sum(precision, \*params): )
* 키워드 언팩 연산자(\*\*)  
  매개변수의 개수를 가변적으로 사용할 수 있도록 함  
  키워드 인자들을 전달해 매개변수를 **딕셔너리** 형식으로 처리함.
* 기본 값을 갖는 매개변수 설정도 가능하다. (일반 매개변수 앞에 놓이지 못한다)  
  (ex> def calc(x, y, operator=”+”): )
* 람다식 🡺 lambda 매개변수 : 반환값  
  (ex> calc(plus, 10, 5) 🡪 calc(lambda a, b: a-b, 10, 5) )
* 클로저: 중첩함수를 반환하는 방법   
  🡪 정보 은닉 가능







예외 처리

예외처리

* 예외가 발생했을 때 처리하지 않으면 프로그램이 끝나게 되므로 안전하게 처리해야 함.
* if문을 이용한 예외처리는 정상적인 흐름을 제어할 경우에만 사용 가능.

예외 객체

* 코드 실행 중 오류가 발생하여 만들어진 것. 오류 발생과 관련한 정보를 가지고 있음.
* 예외처리 실습 부분에 함수로 예외처리 만드는 부분 다시 봐도 좋을 듯. (raise도 사용)

예외 왜 일부러 일으키는 거?

* 이 이유 때문인듯: 예외가 발생했을 때 처리하지 않으면 프로그램이 끝나게 되므로 안전하게 처리해야 함

표준 모듈과 활용

From ~ import ~ 문을 이용한 선택적 로딩

* Math 모듈이 로딩되지 않고 함수와 값이 직접 로딩됨.

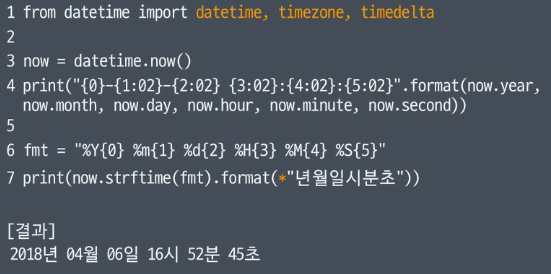
Ex) 

Ramdom 모듈

* random()  
  0.0 <= N < 1.0범위의 부동소수점 난수 N 반환
* uniform(1.0. 10.0)  
  지정된 범위 내의 부동소수점 난수 N 반환
* randrange(start, stop, step) (인자는 range랑 비슷하게 들어가는 것 같다)  
  정수형 난수 N 생성
* choice(data\_list)  
  인자로 전달된 시퀀스 객체의 항목 중 임의 항목 반환
* choices(data\_list, 2)  
  인자로 전달된 시퀀스 객체의 항목 중 임의의 K개 반환 복원 추출 기능을 가진 시뮬레이션 함수
* sample(data\_list, 2)  
  인자로 전달된 시퀀스 객체, 혹은 set 객체 항목 중 임의의 K개 반환, 비복원추출 기능을 가진 시뮬레이션 함수
* shuffle()  
  인자로 전달된 시퀀스 객체의 항목을 뒤섞는 함수. 반환값은 없고 원본 객체의 항목의 순서를 뒤섞음

datetime 모듈

* datetime



(\*: 인자 unpack 연산자)

서드파티 모듈 설치 및 활용 (서트파티 모듈: 외부 모듈)

설치: pip install [모델명]

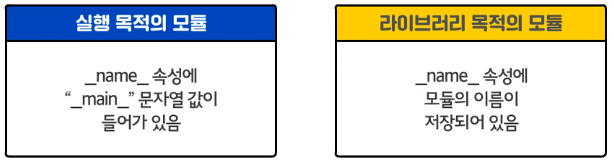
제거: pip uninstall [모델명]

* 영상 -> pytz 모듈 사용법. (timezone)

사용자 정의 모듈

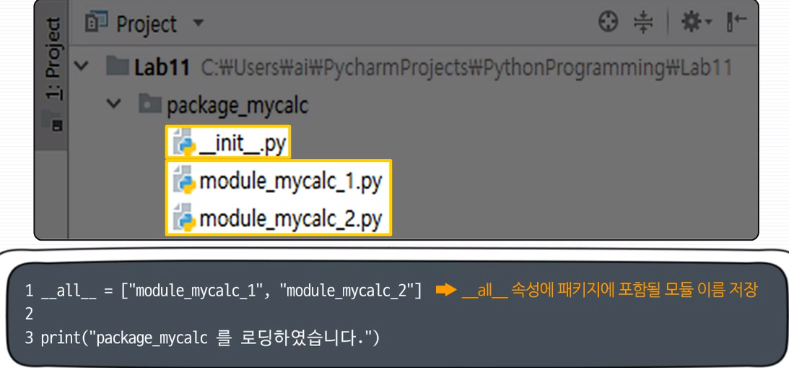
* 크게 라이브러리 모듈과 라이브러리 모듈을 사용하기 위한 실행 모듈로 나뉜다.

모듈의 \_\_name\_\_ 속성



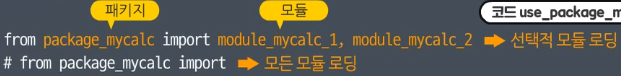
사용자 정의 패키지 (패키지: 모듈을 모아서 하나의 패키지로 만들 수 있다.)

* 패키지를 구성하기 위해서는 먼저 폴더를 생성해야 한다.



패키지 사용

* from 패키지명 import 모듈명



리스트의 생성 및 조작법

* append & extend 차이점  
  🡪 리스트를 인자로 전달 시 append는 리스트 자체가 항목으로 추가, extend는 인자로 전달된 리스트의 내부 항목을 추가.

리스트 요소 삭제하기

* del a[x]  
  x번째 요솟값을 삭제한다.
* pop(x)  
  x번째 요소를 돌려주고 그 요소는 삭제한다.
* remove(x)  
  리스트에서 첫 번째로 나오는 x를 삭제하는 함수.
* clear()  
  모든 항목을 제거해 빈 리스트 객체 생성

리스트 내포

* 반복 가능한 자료형의 경우 리터럴 안에서 for문을 사용하면 내포 기능 사용 가능.

(ex> )

(ex2> )

(ex3> )

(ex4> )

튜플의 생성 및 조작법

튜플의 생성 방법 中 1







* **새로운 튜플 객체 반환**

튜플 내포

(ex> )

(ex2> )

(ex3> )

객체

객체의 생성과 소멸, 그리고 self

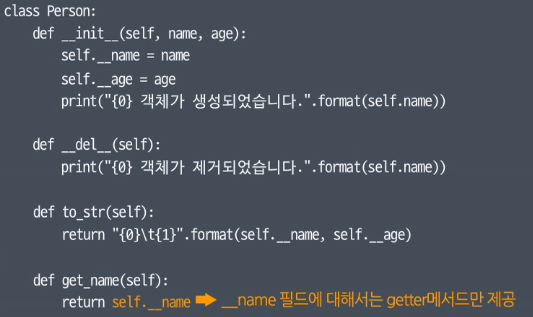
* 객체를 생성하기 위해 호출하는 생성자 메서드  
  \_\_init\_\_ 메서드  
  ex> def \_\_init\_\_(self, name, age):
* 객체가 소멸되기 전에 호출되는 소멸자 메서드  
  \_\_del\_\_ 메서드 (항상 쓰는 건 아님),   
  def \_\_del\_\_(self): 처럼 self만 쓴다.

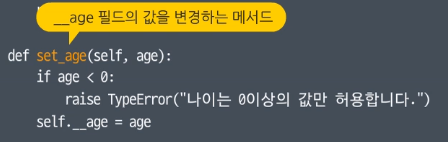
인스턴스 변수-------

접근 제한기능: self.\_\_name = name 🡺 프라이빗 필드 생성  
getter/setter메서드의 제공 여부에 대한 고민 필요.

Q. 왜 이렇게 프라이빗 필드를 만들어야 하는지 정확히는 잘 모르겠다.

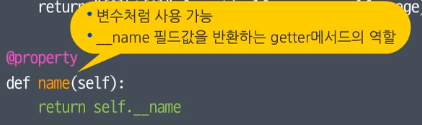
* getter: 멤버를 읽어오는 메서드 / setter: 멤버를 변경하는 메서드

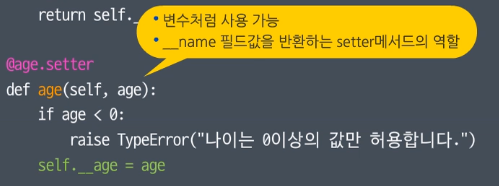




데코레이터 기능: 변수 이름과 같은 메서드를 만들어 사용가능

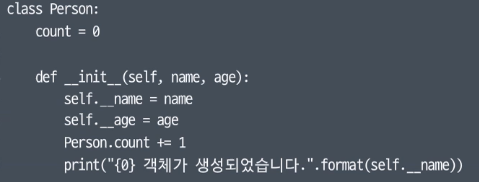
* @property // @property의이름.setter 와 같이 사용가능.







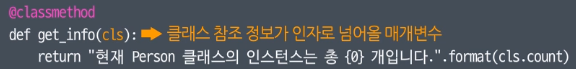
클래스 변수



여기서 count같은 거. 불러올 땐 Person.count와 같이.



클래스 메서드



Q 클래스 메서드 데코레이터 쓰는 이유? 써야만하나?

비교연산자 오버로딩

Q. \_\_gt\_\_, \_\_ge\_\_, \_\_lt\_\_, \_\_le\_\_, \_\_eq\_\_, \_\_ne\_\_는 비교연산자를 나타내는 고유의 메서드 같은 건가?

\_\_str()\_\_메서드

* str()함수에 객체를 전달해 문자열로 변환할 수 있다.

클래스 상속

파이썬에서는 단일 상속만 지원해서 하나의 클래스는 단일 클래스에서만 상속할 수 있다.



repr함수는 객체 출력 시 주로 사용.

Q. \_\_class\_\_의 의미가 뭐지?

* 정렬 (중요, 잘 기억해두면 좋을 듯)



Key는 어떤 기준으로 정렬할지 세팅하는 것.  
Reverse의 값은 기본이 false라서 내용을 넣지 않으면 false로 동작한다.