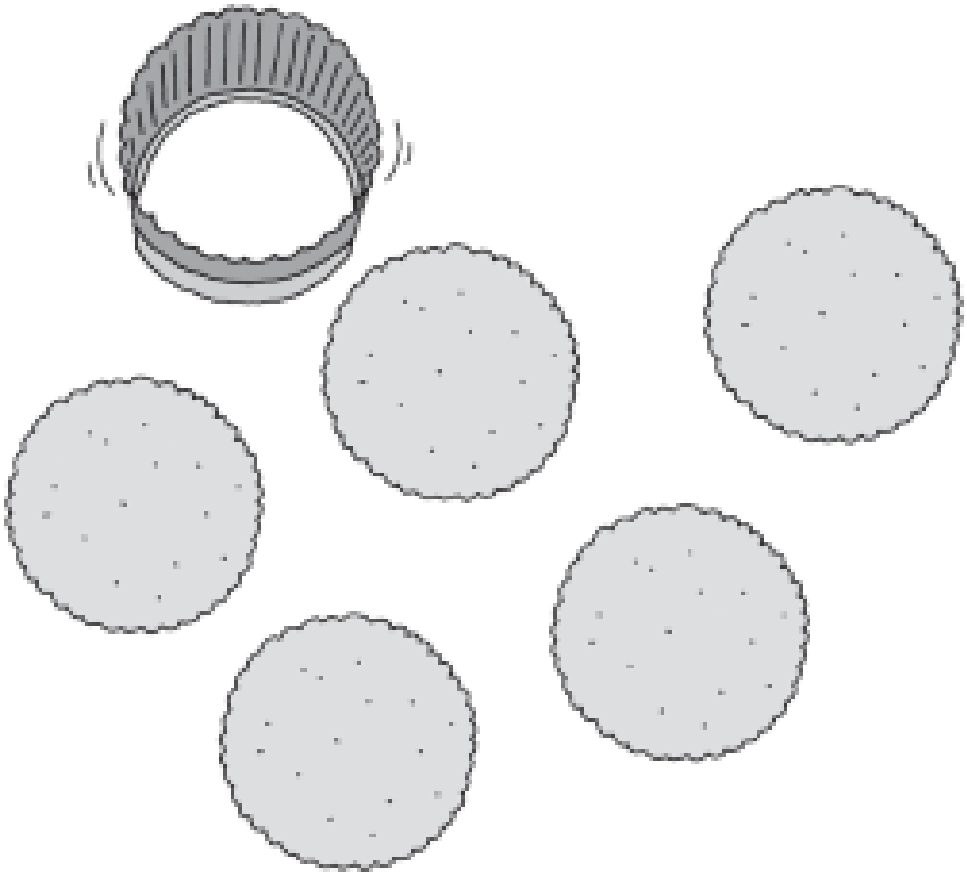
Chapter.01 파이썬 프로그래밍-09. Class

💡 Class : OOP에서 구현하려는 개념을 추상화한 **코드 템플릿**.



Source: https://wikidocs.net/28

- Class를 이용해서 구현하려는 개념을 객체(object)의 형태로 찍어낼 수 있습니다.
- 구현하려는 대상의 특성을 Class variable로, 대상이 수행해야 하는 일을 Class method로 구현해야 합니다.

- Constructor(생성자)를 통해서 객체를 찍어내는 틀을 정의할 수 있습니다.

```
# Python Class example
class Human(superclass):  # 상속을 받고 싶을 때, 상속받을 클래스 이름을 파라미터로 지정.
  def __init__(self, name, weight): # Constructor
    self.name = name
    self.weight = weight
    ...

  def gain_weight(self, a, b):
    tmp_weight = self.weight + a
    <statement>
    ...
    return tmp_weight


>>> object1 = Human("Kim", 70)  # class_name() : __init__ method call
>>> object1.name
>>> "Kim"
>>> object1.gain_weight(5, 7)
>>> 75
```

- 생성자는 __init__() 함수를 이용하여 구현합니다.
- 구현되는 객체는 `self` 라는 자체 변수를 가집니다. self는 말 그대로 객체 자기 자신을 지칭합니다.
- self 변수를 통해서 모든 객체는 자기 자신을 구분할 수 있습니다.
- Class method도 self 변수를 이용하여 객체를 구분합니다.
- self는 Class variable이기 때문에 하나의 Class내에서 통용됩니다.
- Class도 역시 재사용성을 고려하여 디자인되어야 합니다.
- Class로 구현할 때 제일 중요한 포인트는 "**어떤 특성과 어떤 기능을 구현할 것인가**" 입니다.

ML/DL Project 실무에서의 활용.

- Tensorflow/Keras template

https://github.com/Husseinjd/keras-tensorflow-template

- Pytorch template

> https://github.com/victoresque/pytorch-template

Key Points

1. 구현하려는 대상의 특성과 기능을 명확하게 정의합니다.
2. OOP의 개념을 명확하게 하여, 기능별 구현 사항을 명확하게 정의합니다. (명세 사항)