1. `@staticmethod`: This decorator is used to define a static method in a class. It allows you to call the method without creating an instance of the class.

```
python
class MyClass:
    @staticmethod
    def my_static_method():
        print("This is a static method")

# Calling the static method
MyClass.my_static_method()
```

2. `@classmethod`: This decorator is used to define a class method in Python. It receives the class as the first argument instead of an instance of the class.

```
python
class MyClass:
    @classmethod
    def my_class_method(cls):
        print("This is a class method")
        print("Class:", cls)

# Calling the class method
MyClass.my_class_method()
```

3. `@property`: This decorator is used to define a method as a property of a class. It allows you to access the method like an attribute, without using parentheses.

```
python
class Circle:
    def __init__(self, radius):
        self._radius = radius

    @property
    def radius(self):
        return self._radius

    @radius.setter
    def radius(self, value):
        self._radius = value

# Creating an instance and accessing the property
circle = Circle(5)
print(circle.radius)  # Output: 5

# Modifying the property
circle.radius = 10
print(circle.radius)  # Output: 10
```