```
"""
An example of how lambda functions can make attack code more concise.
"""


class Character:
    """
    A class representing a character in a game.
    can be Barbarian, Wizard, Paladin, etc.
    """

    def __init__(self, hp, attack, defend):
        self.hp = hp
        self.attack = attack
        self.defend = defend


"""
----------------------------------------------------------------
Now, let's look at how we deal with damage calculation in a game
WITHOUT using lambda functions.
"""


class DamageCalculator:
    """
    we will need to define a class to represent the different ways
    """
    MINUS_METHOD = 1
    TIMES_METHOD = 2
    TRUE_DAMAGE = 3


class BattleManager:
    # without lambda function we have to switch between different cases
    # there can be a lot of cases in a real game
    def deal_damage(attacker, defender, calculator, args):
        if calculator == DamageCalculator.MINUS_METHOD:
            damage = attacker.attack - defender.defend
        elif calculator == DamageCalculator.TIMES_METHOD:
            damage = round(attacker.attack * (defender.defend *
                           1.0 / (args[0] + defender.defend)))
        elif calculator == DamageCalculator.TRUE_DAMAGE:
            damage = args[0]
        else:
```

```
            damage = 0

        # we will not consider buffs here because it's just an example
        defender.hp -= damage


"""
----------------------------------------------------------------
Now, let's look at how we deal with damage calculation in a game
using lambda functions.
"""

# no class DamageCalculator here anymore


class BattleManagerWithLambda:
    # using lambda function we can make the code more concise
    def deal_damage(attacker, defender, damage_calculator, args):
        if damage_calculator:
            defender.hp -= damage_calculator(attacker, defender, args)


"""
----------------------------------------------------------------
Use those two classes to deal damage
"""


def main():
    attacker = Character(hp=100, attack=50, defend=20)
    defender = Character(hp=100, attack=30, defend=10)
    # ---------------------------------
    # example of not using lambda function
    # deal damage using minus method
    BattleManager.deal_damage(
        attacker, defender, DamageCalculator.MINUS_METHOD, [])
    print(defender.hp)

    # deal damage using times method
    BattleManager.deal_damage(
        attacker, defender, DamageCalculator.TIMES_METHOD, [10])
    print(defender.hp)

    # deal damage using true damage method
    BattleManager.deal_damage(
```

```
        attacker, defender, DamageCalculator.TRUE_DAMAGE, [10])
    print(defender.hp)

    # ----------------------------------
    # example of using lambda function
    attacker = Character(hp=100, attack=50, defend=20)
    defender = Character(hp=100, attack=30, defend=10)

    # deal damage using minus method
    BattleManagerWithLambda.deal_damage(
        attacker, defender,
        lambda attacker, defender, args: attacker.attack - defender.defend,
        []
    )
    print(defender.hp)

    # deal damage using times method
    BattleManagerWithLambda.deal_damage(
        attacker, defender,
        lambda attacker, defender, args: round(
            attacker.attack * (
                defender.defend * 1.0 / (args[0] + defender.defend)
            )
        ),
        [10]
    )
    print(defender.hp)

    # deal damage using true damage method
    BattleManagerWithLambda.deal_damage(
        attacker, defender,
        lambda attacker, defender, args: args[0],
        [10]
    )
    print(defender.hp)


if __name__ == "__main__":
    main()
```