"""

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:

    def \_\_init\_\_(self):

        pass

    # without lambda function we have to switch between different cases

    # there can be a lot of cases in a real game

    def deal\_damage(self, 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:

    def \_\_init\_\_(self):

        pass

    # 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

    battle\_manager = BattleManager()

    # deal damage using minus method

    battle\_manager.deal\_damage(

        attacker, defender, DamageCalculator.MINUS\_METHOD, [])

    # deal damage using times method

    battle\_manager.deal\_damage(

        attacker, defender, DamageCalculator.TIMES\_METHOD, [10])

    # deal damage using true damage method

    battle\_manager.deal\_damage(

        attacker, defender, DamageCalculator.TRUE\_DAMAGE, [10])

    # -----------------------------------

    # example of using lambda function

    battle\_manager\_with\_lambda = BattleManagerWithLambda()

    # deal damage using minus method

    battle\_manager\_with\_lambda.deal\_damage(

        attacker, defender,

        lambda attacker, defender, args: attacker.attack - defender.defend,

        []

    )

    # deal damage using times method

    battle\_manager\_with\_lambda.deal\_damage(

        attacker, defender,

        lambda attacker, defender, args: round(

            attacker.attack \* (

                defender.defend \* 1.0 / (args[0] + defender.defend)

            )

        ),

        [10]

    )

    # deal damage using true damage method

    battle\_manager\_with\_lambda.deal\_damage(

        attacker, defender,

        lambda attacker, defender, args: args[0],

        [10]

    )