```
__author__ = 'Kai'


# ---- Initial Comments and Information ----


# ---- Importing Functions ----
import random

# ---- Temporary Stored Variables ----


# ---- Defined Functions ----
class warrior:
    def __init__(self, name, fighting_style):
        """
        Base self class for the warriors.
        :param name: name of the warrior, eg Ninja/Pirate
        :param fighting_style: style for the warrior, eg ninja_style
        """
        self.name = name
        self.fighting_style = fighting_style
        self.weapon_list = []
        self.win_score = 0

    def acquire_weapon(self):
        """
        Used to grab a weapon from one of the stashes, and giving it to a warrior.
        :return: none
        """
        Random_Hands = random.randint(0,1)
        if Random_Hands == 0:
            length_stash = len(self.fighting_style.one_handed_list) - 1
            random_stash_1 = random.randint(0, length_stash)
            self.weapon_list.append(self.fighting_style.one_handed_list[random_stash_1])

self.fighting_style.one_handed_list.remove(self.fighting_style.one_handed_list[random_stash_1])

            length_stash = len(self.fighting_style.one_handed_list) - 1
            random_stash_2 = random.randint(0, length_stash)
            self.weapon_list.append(self.fighting_style.one_handed_list[random_stash_2])

self.fighting_style.one_handed_list.remove(self.fighting_style.one_handed_list[random_stash_2])
        elif Random_Hands == 1:
            length_stash = len(self.fighting_style.two_handed_list) - 1
            random_stash = random.randint(0, length_stash)
            self.weapon_list.append(self.fighting_style.two_handed_list[random_stash])

    def information(self):
        """
        Used to display what weapons a warrior is wielding, eventually may be added to
        :return: print string
        """
        j = ''
        for i in self.weapon_list:
            j = j + ', ' + i.name
        return (self.name, 'is wielding:{0}'.format(j))

    def attack_information(self):
        """
        Used to display what weapons a warrior is wielding, eventually may be added to
        :return: print string
        """
        total = 0
        for i in self.weapon_list:
            total += int(i.attack_value())
        return total

    def defend_information(self):
        """
        Used to display what weapons a warrior is wielding, eventually may be added to
        :return: print string
        """
        total = 0
        for i in self.weapon_list:
            total += int(i.defence_value())
        return total

    # def parry_result(self):
    #     """
    #
    #     :return:
    #     """
    #     parry_list = []
    #     for i in self.weapon_list:
    #         parry_list.append(str(i.Parry()))
    #     print(parry_list[0], parry_list[1])
    #     if parry_list[0] or parry_list[1] == True:
    #         return True
    #     else:
    #         return False

class fighting_style:
    def __init__(self, style):
        """
        choosing which style to use for a warrior
        :param style: Pirate_style or Ninja_style
        """
        self.style = style
        self.one_handed_list = []
        self.two_handed_list = []

    def deposit_weapon(self, weapon):
        """
        used to put weapons into a stash for later use
        :param weapon: the weapon's name
        :return: none
        """
        if weapon.TH == False:
            self.one_handed_list.append(weapon)
        elif weapon.TH == True:
            self.two_handed_list.append(weapon)

class weapon:
    def __init__(self, name, attack_value, defence_value, parry, two_handed):
        """
        Used to hold values for a weapon
        :param name: name of a weapon
        :param attack_value: attack points integer
        :param defence_value: defence points integer
        :parry: boolean value for if a weapon can parry or not
        :two_handed: boolean value for if a weapon is two handed
        """
        self.name = name
        self.AV = attack_value
        self.DV = defence_value
        self.Parry = parry
        self.TH = two_handed

    def attack_value(self):
        """
        grabs the attack value of a weapon
        :return: AV (attack_value)
        """
        return(self.AV)

    def defence_value(self):
        """
        grabs the defence value of a weapon
        :return: DV (defence_value)
        """
        return(self.DV)

class combat:
    def __init__(self, warrior_1, warrior_2):
        """
        Used to have two warriors fight, using weapons, user input is how many total battles, each battle
consists of a random amount of duels from 5 to 10.
        :param warrior_1: define the first warrior, eg Pirate
        :param warrior_2: define the second warrior, eg Ninja
        :return: none
        """
        user_input = int(input('Input number of battles as an integer: '))
        tie_score = 0

        while user_input > 0:
            coin_flip = random.randint(0, 1)
            if coin_flip == 0:
                attacker = warrior_1
                defender = warrior_2
            elif coin_flip == 1:
                attacker = warrior_2
                defender = warrior_1
            else:
                print('Error!! Combat Coin flip not 0 or 1!')

            Pirate_Style.one_handed_list.clear()
            Ninja_Style.one_handed_list.clear()

            Pirate_Style.deposit_weapon(Cutlass)
            Pirate_Style.deposit_weapon(Pistol)
            Pirate_Style.deposit_weapon(Boomstick)

            Ninja_Style.deposit_weapon(Throwing_Star)
            Ninja_Style.deposit_weapon(Short_Blade)
            Ninja_Style.deposit_weapon(Katana)

            warrior_1.acquire_weapon()
            warrior_2.acquire_weapon()

            print('The attacker, the', warrior.information(attacker))
            print('The defender, the', warrior.information(defender))

            duel_amount = random.randint(5, 11)
            while duel_amount > 0:

                # if defender.parry_result() == True:
                #     if random.randint(1, 2) == 1:
                #         print('Parried!')
                #         defender.win_score += 1
                #     else:
                #         if attacker.attack_information() > defender.defend_information():
                #             attacker.win_score += 1
                #         elif attacker.attack_value() < defender.defend_informatione():
                #             defender.win_score += 1
                #        elif attacker.attack_value() == defender.defend_information():
                #            tie_score += 1
                #        else:
                #            print('Error!! Combat Comparison failure!')
                # else:
                if attacker.attack_information() > defender.defend_information():
                    attacker.win_score += 1
                elif attacker.attack_value() < defender.defend_informatione():
                    defender.win_score += 1
                elif attacker.attack_value() == defender.defend_information():
                    tie_score += 1
                else:
                    print('Error!! Combat Comparison failure!')
                duel_amount -= 1

            warrior_1.weapon_list.clear()
            warrior_2.weapon_list.clear()
            user_input -= 1
        if attacker.win_score > defender.win_score:
            print()
            print('The Winner is:', attacker.name, 'With a score of', attacker.win_score, 'to',
defender.win_score, 'and', tie_score, 'ties!')
        elif attacker.win_score < defender.win_score:
            print('The Winner is:', defender.name, 'With a score of', defender.win_score, 'to',
attacker.win_score, 'and', tie_score, 'ties!')
        else:
            print("Woah! It's a tie!?", tie_score)


# ---- Content Code ----
Pirate_Style = fighting_style('Pirate')
Ninja_Style = fighting_style('Ninja')

Pirate = warrior('Pirate', Pirate_Style)
Ninja = warrior('Ninja', Ninja_Style)

Cutlass = weapon('Cutlass', '2', '1', True, False)
Pistol = weapon('Pistol', '3', '0', False, False)
Boomstick = weapon('Boomstick', '5', '0', False, True)

Throwing_Star = weapon('Throwing Star', '3', '0', False, False)
Short_Blade = weapon('Short_Blade', '1', '3', False, False)
Katana = weapon('Katana', '3', '4', True, True)

Pirate.information()
Ninja.information()
combat(Pirate, Ninja)
# ---- End of Documentation ----
```