* **Day One**- by the end of the first day:
  + sketch out the character class

create the class Character, parameter is object

define initial function with parameter self, name = “Monster” (as auto since the code provided changes it to Hero, hitPoints = 10, hitChance = 50, maxDamage = 5, armor = 2:

then do the super() init thing

self.name = name, self.hitPoints = hitPoints, self.hitChance = hitChance, self.maxDamage = maxDamage, self.armor = armor

define the properties of the private attributes 🡪

@property

def name(self)

return self.\_\_name – this will retrieve name from private attribute- format correct

@name.setter- def name(self, value): self.\_\_name = value

CONTINUE FOR EACH OF BELOW PRIVATE ATTRIBUTES

name,

hitPoints, (must be an integer) (must be able to go to 0 or negative to end game)

hitChance, (must be integer, zero to 100)

maxDamage, (positive integer)

armor (positive integer)

def testInt(self, value, min = 0, max = 100, default = 0):  
    """ takes in value   
        checks to see if it is an int between  
        min and max.  If it is not a legal value  
        set it to default """  
  
    out = default  
  
    if type(value) == int:  
        if value >= min:  
            if value <= max:  
                out = value   
            else:  
                print("Too large")  
        else:  
            print("Too small")  
    else:  
        print("Must be an int")  
  
    return out

* + Design and write the printStats() method (function in a class)

Def printStats():

Print {name} (then lines for formatting)

Print Hit points: {hitPoints}

Print Hit chance: {hitChance}

Print Max Damage: {maxDamage}

Print Armor: {armor}

* + In main, instantiate a class and print its stats
  + stretch goal - implement filters in properties as needed
* **Day Two**
  + Modify properties as needed to add filters- done
  + Design the algorithm for the hit method

Import random

Define hit- parameter is character

hitInt = random.randrange between 0-100

if hitInt is between 0 and {hitChance} then you scored a hit

print “{character} hits {other character}…”

DamageDone = random.randrange between 1 and {maxDamage}

If {other character} armor > 0:

Then DamageDone = 0

{other character} armor = int({other character} armor)

{other character} armor = {other character} armor – 1

Print “{other character}’s armor absorbed the hit”

Print “{other character” has {other character’s armor} armor left”

Else:

{other character} hitPoints = int({other character} hitPoints)

{other character} hitPoints = {other character} hitPoints – DamageDone

Print “{other character} has no armor left”

Print “ {other character} loses {DamageDone} points”

else you did not hit the other character

print “You missed” or something

SOMEWHERE AT THE END- IF HITPOINTS <= 0 then game over

* + Implement the hit method
  + Modify main() to incorporate one or more rounds of combat
  + stretch goal - design and implement the fight
* **Day Three CURRENT SPOT**
  + Design the fight algorithm

Define fight function- parameters monster and hero

keepGoing = True

while keepGoing:

Monster gets the output of heroHit

Print “Monster: {monster.hitPoints} HP “

Hero gets the output of MonsterHit

Print “Hero: {hero.hitPoints} HP “ (spacing after this)

Answer = Input of “press for another round”

If answer == “”:

keepGoing gets True

if hero.hitPoints <= 0:

keepGoing gets False

print “Monster wins!”

if monster.hitPoints <= 0:

keepGoing gets False

print “Hero wins!”

else:

keepGoing = True

else:

keepGoing gets True

if hero.hitPoints <= 0:

keepGoing gets False

print “Monster wins!”

if monster.hitPoints <= 0:

keepGoing gets False

print “Hero wins!”

else:

keepGoing = True

* + Implement the fight function (taking two characters as parameters)
  + Build combat.py to incorporate everything in the module
  + Test everything
  + Stretch goal - add some blackbelt features