2a. For my Create Task I have chosen to use the coding language Python. To be more specific, I am using Python 2.7.10. The purpose of the program I have created is to simulate an RPG, or role-playing game. The video that I have made of my program will demonstrate a small part of my code, which is the introduction and the beginning part of attacking the enemy.

2b. While creating my program, I encountered a lot of errors. Most of them resided in the usage of global variables, variables that “stretch” across the whole program instead of being ingrained into a specific function like local variables. However, while figuring out how to solve my problem for the global variables, I discovered something. To allow a global variable to be changed, and not just referenced, you must call globalize that variable again so that the computer can know it was changed. Other than the global issue, nothing else went wrong with the developmental process of the program. Mostly due to the usage of separate definitions, if else commands, and print statements.

2c.
```
if current_action == 1:
    print "You choose to attack the {}.\n".format(enemy_name)
    player_random_attack = random.randint(player_stats[0] - 3, player_stats[0] + 3)
    enemy_random_attack = random.randint(enemy_stats[0] - 3, enemy_stats[0] + 3)
    print "You attack first dealing {} damage to the {}.\n".format(player_random_attack,
enemy_name)
    enemy_stats[1] -= player_random_attack
    if enemy_stats[1] <= 0:
      print "The {} has died. You win! Congratulations!!\n".format(enemy_name)
      player_stats[3] += enemy_stats[2]
      if level_stats[1] == 0 and level_stats[0] == 0 and player_stats[3] >= 100:
        print "Congratulations!! You have now levelled up to level 2"
        player_stats[2] = 2
        level_stats[0] = 1
      elif level_stats[0] == 1 and level_stats[1] == 0 and player_stats[3] >= 250:
        print "Congratulations!!!! You are now at the current max level! You are now level 3"
        player_stats[2] = 3
        level_stats[1] = 1
      replay_choose = raw_input("Type 'replay' if you would like to continue playing with current
stats: \n")
      if replay_choose == "replay":
        print "You have chosen to replay the game\n"
        replay()
      else:
        print "Are you sure you want to restart? You're at level {}. Type 'replay' to continue with
your current stats.\n".format(player_stats[2])
        replay_choose_again = raw_input()
        if replay_choose_again == "replay":
          print "You have chosen to replay the game."
```

```
            replay()
          else:
            print "I guess you're not gonna keep playing. GAME OVER. All stats will be reset to level
one once you hit run."
      else:
        print "The {} has {} HP left.\n".format(enemy_name, enemy_stats[1])
        print "The {} retaliates dealing {} damage to you.\n".format(enemy_name,
enemy_random_attack)
        player_stats[1] -= enemy_random_attack
        if player_stats[1] <= 0:
          print "You have died. The {} has won. GAME OVER. Press run to play
again".format(enemy_name)
        else:
          print "You have {} HP left.".format(player_stats[1])
          time.sleep(3)
          line()
          attack_start()
```

Within my created algorithm, I used many if else commands to determine whether or not the player's health or enemy's health was below 0. Each command however, can function on it's own provided that you give each variable a value to use for said command. The initial if else command is deciding if the player input the number 1, which stands for the attack simulation. All of the other if else commands inside of it are determining who has more or less than zero health. Since the player attacks first in the game, I make sure to read the player's health at the beginning, ensuring the player is still alive enough to attack. Second I read the enemy's health after the attack to decide if it has died or it should continue fighting. Last I re-check the player's health to make sure they haven't died from the enemy's counterattack. All of these together create the attack part of the simulation you see above you.

2d.
```
global player_stats
global penalty_damage
global level_stats
global current_max_HP
player_stats = [10, 30, 1, 0] #[attack damage, health points, level, exp]
level_stats = [0, 0] #0 equals false and 1 equals true
penalty_damage = 10
```

My abstraction within my program is right above this text, and its use is to reduce the strain and complexity upon the computer and also upon my brain whilst coding. In this segment of code, I use lists to show most of my information. In the beginning of programming I had each part of each list separate and it would be a hassle to globalize each individual variable. To make this easier I created a specific list for each of the general elements in my game. I use player_stats

for all of my player's stats, and the level_stats are a separate variable used as a checkpoint when you level up.

3.

```
import random
import time

def line():
  print "\n----------------------------"

def intro():
    global player_stats
    global penalty_damage
    global level_stats
    global current_max_HP
    player_stats = [10, 30, 1, 0] #[attack damage,
  health points, level, exp]
    level_stats = [0, 0] #0 equals false and 1
  equals true
    penalty_damage = 10
  starting_point = raw_input("To start this simulation, please type start: \n")
  if starting_point == "start":
    name_choose()
  else:
    print "Don't be a wiseass. Type start correctly."
    intro()

def name_choose():
  global player_name
  player_name = raw_input("\nTo continue, what would you like to be called?: \n")
  print "\nWell {}, I am glad to have you try my new simulation. I hope you have more fun than I
did trying to make this piece of garbage.\n".format(player_name)
  simulation_start()

def simulation_start():
  global player_name
```

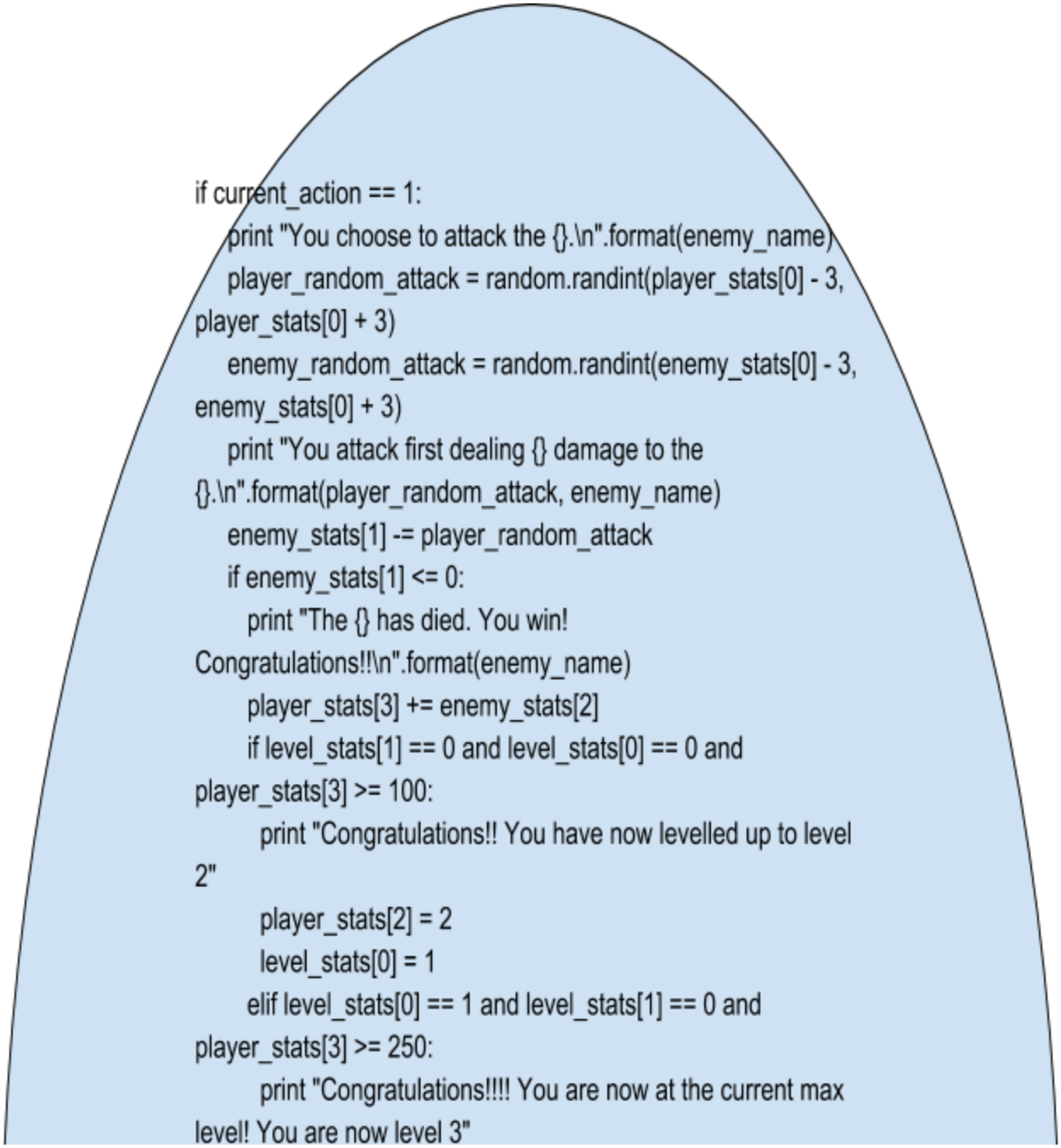
```
  global player_stats
  print "Your starting stats are {} attack and {} HP (Health Points). You can fight enemies until
your HP runs out, until the enemy dies, or you can flee and end the game. Have
fun!\n".format(player_stats[0], player_stats[1])
  start_game = raw_input("Type 'go' to start the game, or type 'again' to retype your
name.\n\nSidenote: When you go back you will have to see the stats and everything again.\n")
  if start_game == "go" or start_game == "Go":
    print "You started the game!"
    enemy_choose()
  elif start_game == "again" or start_game == "Again":
    name_choose()
  else:
    print "Guess what! You now go back even though you typed something different. Happy?\n"
    name_choose()

def enemy_choose(): #When I make the replay feature, it will direct here.
  global enemy_name
  global enemy_stats
  enemy_name = random.choice(["Floating Hat", "Unicorn", "Skeleton", "Giant Rat"])
  #all enemy_stats are formatted as [attack damage, health points, experience]
  #Later on, experience points per enemy will vary. For now they will stay the same
  if enemy_name == "Floating Hat":
    enemy_stats = [3, 25, 100]
  elif enemy_name == "Unicorn":
    enemy_stats = [9, 35, 100]
  elif enemy_name == "Skeleton":
    enemy_stats = [11, 17, 100]
  elif enemy_name == "Giant Rat":
    enemy_stats = [6, 31, 100]
  intro_enemy_text()

def intro_enemy_text():
  print "\nA(n) {} appears in front of you".format(enemy_name)
  attack_start()

def attack_start():
  global player_stats
  global enemy_stats
  global level_stats
  global current_max_HP
  if player_stats[1] > 0:
   print "\nThe {} stays and wants to continue fighting. What will you do?\n\nType the number you
want to choose\n\n1. Attack  2. Block  3. Flee\n".format(enemy_name)
```

```
  current_action = input()
```



```
if current_action == 1:
    print "You choose to attack the {}.\n".format(enemy_name)
    player_random_attack = random.randint(player_stats[0] - 3,
player_stats[0] + 3)
    enemy_random_attack = random.randint(enemy_stats[0] - 3,
enemy_stats[0] + 3)
    print "You attack first dealing {} damage to the
{}.\n".format(player_random_attack, enemy_name)
    enemy_stats[1] -= player_random_attack
    if enemy_stats[1] <= 0:
      print "The {} has died. You win!
Congratulations!!\n".format(enemy_name)
      player_stats[3] += enemy_stats[2]
      if level_stats[1] == 0 and level_stats[0] == 0 and
player_stats[3] >= 100:
        print "Congratulations!! You have now levelled up to level
2"
        player_stats[2] = 2
        level_stats[0] = 1
      elif level_stats[0] == 1 and level_stats[1] == 0 and
player_stats[3] >= 250:
        print "Congratulations!!!! You are now at the current max
level! You are now level 3"
  elif current_action == 2:
    print "You choose to defend against the {}.\n".format(enemy_name)
    heal_chance = random.randint(1, 4)
    if heal_chance == 2:
      print "While blocking against the {}, you feel confident in your abilities to win and gain 5
health back.\n".format(enemy_name)
      player_stats[1] += 5
```

```
      if player_stats[1] > current_max_HP: #Set these numbers equal to the maximum HP
amount
        player_stats[1] = current_max_HP
      print "Your HP is now at {}\n".format(player_stats[1])
      time.sleep(3)
      line()
      attack_start()
    else:
      print "You felt the hard attack of the {} and stagger. But you get back up and continue the
fight with nothing different than before.".format(enemy_name)
      time.sleep(3)
      line()
      attack_start()
  elif current_action == 3:
    print "You choose to end the game."
  else:
    global penalty_damage
    print "Try again and actually choose something on the list! You just lost {} health points
because you can't type. Good job.\n".format(penalty_damage)
    player_stats[1] -= penalty_damage
    print "Your current health is {}\n".format(player_stats[1])
    time.sleep(3)
    line()
    attack_start()
 else:
   print "The {} has killed you. Game over for you. Type replay to try again with a new enemy. Or
maybe not it's on a random choice.".format(enemy_name)

def replay():
 global player_stats
 global current_max_HP
 if player_stats[2] == 1: #Checking which level you are before saying what your current stats
are so they can change.
   player_stats[0] = 10
   player_stats[1] = 30
   current_max_HP = 30
 elif player_stats[2] == 2:
   player_stats[0] = 13
   player_stats[1] = 34
   current_max_HP = 34
 elif player_stats[2] == 3:
   player_stats[0] = 15
   player_stats[1] = 40
```

```
    current_max_HP = 40
  line()
  simulation_start()

#Add new variables like def and spd. (defense and speed)
#Add a list of attack moves with varying damage after completing all stats and other minor tasks

intro()
```