Holder.py, the main program

"""  
author: Nicholas Atkins Due Date: 20 Mar 2018  
The main file that holds the program and imports the semaphore class object  
How to run: Ran this using the Pycharm compiler which runs off the standard Idle for python,  
 Using this file you use run and the rest speaks for itself  
Problems: you must declare the variables as global when used inside of a method in python for syntactical reasons  
 Didn't like the idea of only having one semaphore so I included another for clarity, can easily be removed  
"""  
  
from Semaphore import Semaphore # importing the class Semaphore  
  
  
def wait(sem):  
 """  
 created first, looking at the semantics of the xinu wait function  
 :param sem: Semaphore object  
 :return: void  
 """  
 sem.value -= 1  
  
  
def signal(sem):  
 """  
 created second, looking at the semantics of the xinu signal function  
 :param sem: Semaphore object  
 :return: void  
 """  
 sem.value += 1

def proc1(sem1, sem2):  
 """  
 Uses two global variables to  
 :param sem1: Semaphore object  
 :param sem2: Semaphore object  
 :return: void  
 """  
 global n  
 global full  
  
 if(n == len(listy)): # triggers when the list is full  
 full += 1 # So it wont be 0 and will trigger at the end of this function  
 else:  
 wait(sem1) # calls the wait func to lower the val of sem1/part1  
 print('Proc1 has waited')  
 print('Attempting to set position: ', n)  
 listy[n] = n # Does what the above print statement says  
 n += 1 # Increments the global variable  
 print('Incremented n to: ', n)  
 print('Signal sent from proc1')  
 signal(sem2)  
 print() # Fresh line for ascetic

def proc2(sem1, sem2):  
 """  
 uses two global variables n and full to that are shared  
  
 :param sem1: Semaphore object  
 :param sem2: Semaphore object  
 :return: void  
 """  
 global n  
 global full  
  
 if(n == len(listy)): # triggers when the list is full  
 full += 1 # So it wont be 0 and will trigger at the end of this function  
 else:  
 wait(sem2) # calls the wait func to lower the val of sem1/part1  
 print('Proc2 has waited')  
 print('Attempting to set position: ', n)  
 listy[n] = n # Doing what the print statement above intended to do  
 n += 1 # increments the global variable  
 print('Incremented n to: ', n)  
 print('Signal sent from proc2')  
 signal(sem1)  
 print() # Fresh line for ascetic  
  
  
listy = [0, 0, 0, 0, 0, 0, 0, 0] # global list of 8 vars, to make sure all are different  
  
n = 0 # shared resource used between proc1 and proc2  
  
full = 0 # To detect when the resources are used up  
part1 = Semaphore(1) # Much nicer to have two  
part2 = Semaphore(0) # These act like the producer and consumer model from class

"""  
This is code that I used to make sure the wait and signal did what they were needed to do  
  
print(semmy.value)  
wait(semmy)  
print(semmy.value)  
signal(semmy)  
print(semmy.value)  
"""  
  
# This is a loop that keeps looping until the list is full  
while(full == 0):  
 """  
 This loop is set up so it only works for 2 process, as is the requirement of the assignment  
 """  
 if(part1):  
 # calls process 1  
 proc1(part1, part2)  
 elif(part2):  
 # calls process 2  
 proc2(part1, part2)  
 else:  
 # takes charge and sets signal if both are waiting (just because  
 print('Waiting on a signal')  
 part1.value = 1  
 if(full == 0):  
 print(listy)  
 print() # Fresh line for ascetic  
 else:  
 print('The list has been filled stopping all waiting processes')  
 print('<(0\_0<) (╯°□°）╯︵ ┻━┻ (>0.0)>')  
 print('Final result: ')  
 print(listy) # printing the final result

Semaphore.py, home of the class object

"""  
author: nicholas Atkins Due date: 20th Mar 2018  
This is an object that hold an int value and is used as a signal in the main Holder.py program  
This is not the place where you run the program, information is in the Holder.py comments at the top  
"""  
  
class Semaphore:  
 """  
 This is an object that given a int value will be able to be modified  
 """  
 def \_\_init\_\_(self, val):  
 """  
 given an integer it will initialize the semaphore  
  
 :param val: integer value for the semaphore  
 """  
 self.value = val