import datetime

class Person(object):
    def __init__(self, name):
        #create a person with name name
        self.name = name
        try:
            firstBlank = name.rindex(' ')
            self.lastName = name[firstBlank+1:]
        except:
            self.lastName = name
        self.birthday = None
    def getLastName(self):
        #return self's last name
        return self.lastName
    def setBirthday(self, birthDate):
        #assumes birthDate is of type datetime.date
        #sets self's birthday to birthDate
        self.birthday = birthDate
    def getAge(self):
        #assumes that self's birthday has been set
        #returns self's current age in days
        if self.birthday == None:
            raise ValueError
        return (datetime.date.today() - self.birthday).days
    def __lt__(self, other):
        #return True if self's name is lexicographically less
        #than other's name, and False otherwise
        if self.lastName == other.lastName:
            return self.name < other.name
        return self.lastName < other.lastName
    def __str__(self):
        #return self's name
        return self.name

class MITPerson(Person):
    nextIdNum = 0 #identification number
    def __init__(self, name):
        Person.__init__(self, name)
        self.idNum = MITPerson.nextIdNum
        MITPerson.nextIdNum += 1
    def getIdNum(self):
        return self.idNum
    def __lt__(self, other):
        return self.idNum < other.idNum

class Student(MITPerson):

def getLastName(self):
    #return self's last name
    return self.lastName

def setBirthday(self, birthDate):
    #assumes birthDate is of type datetime.date
    #sets self's birthday to birthDate
    self.birthday = birthDate

def getAge(self):
    #assumes that self's birthday has been set
    #returns self's current age in days
    if self.birthday == None:
        raise ValueError
    return (datetime.date.today() - self.birthday).days

def __lt__(self, other):
    #return True if self's name is lexicographically less
    #than other's name, and False otherwise
    if self.lastName == other.lastName:
        return self.name < other.name
    return self.lastName < other.lastName

def __str__(self):
    #return self's name
    return self.name
class Student(MITPerson):
    pass

class UG(Student):
    def __init__(self, name, classYear):
        MITPerson.__init__(self, name)
        self.year = classYear
    def getClass(self):
        return self.year

class G(Student):
    pass

def ratWorthy(p):
    '''Assumes p is a person
    Returns True iff p is an MIT student'''
    return isinstance(p, Student)

def f(n):
    '''assumes n an int >= 0'''
    answer = 1
    while n > 1:
        answer *= n
        n -= 1
    return answer

def fact(n):
    '''assumes n an int >= 0'''
    if n <= 1:
        return n
    else:
        return n*fact(n - 1)

def g(n):
    '''assumes n an int >= 0'''
    x = 0
    for i in range(n):
        for j in range(n):
            x += 1
    return x

def h(x):
    '''assumes n an int >= 0'''
    answer = 0
    s = str(x)
    for c in s:
        answer += int(c)
    return answer