```
class Item:
  def __init__(self,description:str,movable:bool,extDescription:str):
    self.description = description
    self.movable = movable
    self.extDescription = extDescription

class Location:
  def __init__(self, description:str, items:list):
    self.description = description
    self.items = items
    self.north = None
    self.east = None
    self.south = None
    self.west = None

class World:
  def __init__(self):
    diningroom = Location('in a cheerful dining room',
                            [Item('keys', True, 'The keys jingle as you examine them.'),
                            Item('a small child', True, "The child appears to be pulling CD's out
 of the stack.")])
    kitchen = Location('in a warm, inviting kitchen',
                            [Item('a white microwave', True, "Mmmm... something inside smel
ls yummy!"),
                            Item('a large refrigerator', False, "There is a large padlock around t
he refrigerator.")])
    livingroom = Location('in a comfortable living room',
                            [Item('a grand piano', False, 'The piano is dusty.'),
                            Item('a book of Bach preludes', True, 'Someone has marked up one
of the preludes using a blue marker.')])
    mbedroom = Location('in the master bedroom',
                            [Item('an alarm clock', True, 'The clock is set for 3 a.m.'),
                            Item('a small crib', False, 'The crib is empty.')])
    hallway = Location('in a small hallway', [])
    cbedroom = Location('in another bedroom',
                            [Item('a well-worn teddy bear', True, 'One of the ears is missing.'
)])
    bathroom = Location('in a small bathroom',
                            [Item('a gold key', True, 'The key is glowing.')])

    # Set up exits
    diningroom.west = kitchen
    diningroom.east = livingroom
    diningroom.south = hallway

    kitchen.east = diningroom

    livingroom.west = diningroom
    livingroom.south = hallway

    mbedroom.east = hallway
    mbedroom.north = kitchen

    hallway.west = mbedroom
    hallway.east = cbedroom
    hallway.north = diningroom
    hallway.south = bathroom

    cbedroom.west = hallway

    bathroom.north = hallway

    self.loc = diningroom
    self.inventory = [Item('a green basket', True, 'The basket has a broken handle.')]


  def go(self,dir:str) -> str:
    error = "You can't go that way."
    if dir == 'n':
      if self.loc.north == None:
```

```
        return(error)
      self.loc = self.loc.north
    elif dir == 'e':
      if self.loc.east == None:
        return(error)
      self.loc = self.loc.east
    elif dir == 's':
      if self.loc.south == None:
        return(error)
      self.loc = self.loc.south
    elif dir == 'w':
      if self.loc.west == None:
        return(error)
      self.loc = self.loc.west
    return (self.look())

  def look(self) -> str:
    seen = "You are {}.".format(self.loc.description)
    return(seen)

  def carrying(self) -> str:
    if self.inventory != []:
      itemlist = 'You are carrying:'
      for i in self.inventory:
        itemlist = itemlist +' '+i.description + ","
      itemlist = itemlist.rstrip(',')
      itemlist += '.'
      return(itemlist)
    return("You aren't carrying anything.")

  def lookaround(self)-> str:
    visitems = 'You see:'
    for i in self.loc.items:
      visitems = visitems+' '+i.description+','
    visitems = visitems.rstrip(',')
    visitems += '.'
    return visitems

  def examine(self,descr:str) -> str:
    if self.inventory != []:
      for i in self.inventory:
        if descr in i.description:
          return i.extDescription
    for i in self.loc.items:
      if descr in i.description:
        return i.extDescription
    return ('There is no ' + descr + ' here.')


  def take(self,descr:str) -> str:
    for i in self.loc.items:
      if descr in i.description:
        if i.movable:
          self.inventory.append(i)
          self.loc.items.remove(i)
          return ('You pick up ' + i.description + '.' + self.carrying())
        return ("You can't take that!")
    return ('There is no '+ descr + ' here.')

  def drop(self,descr:str) -> str:
    for i in self.inventory:
      if descr in i.description:
          self.loc.items.append(i)
          self.inventory.remove(i)
          return ('You drop ' + i.description + '.' + self.carrying())
    return ('You are not carrying '+ descr + '.')
```

```python
class Item:
  def __init__(self,description:str,movable:bool,extDescription:str):
    self.description = description
    self.movable = movable
    self.extDescription = extDescription

class Location:
  def __init__(self, description:str, items:list):
    self.description = description
    self.items = items
    self.north = None
    self.east = None
    self.south = None
    self.west = None


class World:
  def __init__(self):
    entrance = Location('in a grand entrance room',[Item('a welcome mat',False,'So fancy, yet so dusty.')])

    hall = Location('in the main hall',[Item('set of armor',False,'Appears to be 16th century genuine chainmail.')])

    lounge = Location('in a room with a warm fireplace',[Item('fire poker',True,'If you touch it, your hands will be covered in soot.')])

    diningroom = Location('in the ornrate dining room',[Item('Solid oak dining table',False,'beautifully handcarved with elaborate designs'),Item('plate of food',True,'A plate filled with turkey, mashed potatoes and other Thanksgiving specialties.')])

    kitchen = Location('in a sparkling clean kitchen that smells like heaven',[Item('silver fork',True,'makes eating a plate of food a little easier'),Item('a cute puppy',True,'The puppy is sniffing the air and looking for food')])

    ballroom = Location('in an elegant ballroom',[Item('large pipes',False,'They seem to be connected to the study...')])

    mbedroom = Location('in an exquisitely furnished bedroom',[Item('a very sturdy safe',False,'has a keyhole with a familiar pattern')])

    library = Location('in a cozy room full of books',[Item('a golden key',True,'looks like you might be able to unlock something with it'),Item('a well-worn book',True,'Looks like a great story. Too bad the last page is missing.')])

    study = Location('in the study room',[Item('large pipe organ',False,'The organ takes up almost the entire room.')])


    # Set up exits
    entrance.west = lounge
    entrance.east = study
    entrance.north = hall

    lounge.east = entrance

    study.west = entrance

    diningroom.east = hall
    diningroom.north = kitchen

    hall.west = diningroom
    hall.east = library
    hall.north = ballroom
    hall.south = entrance

    library.west = hall
    library.north = mbedroom

    kitchen.south = diningroom
```

```python
    ballroom.south = hall
    ballroom.east = mbedroom

    mbedroom.west = ballroom
    mbedroom.south = library




    self.loc = entrance
    self.inventory = [Item('letter',True,'The letter says: Welcome to the mansion! Feel free to look around and explore!')]


  def go(self,dir:str) -> str:
    error = "You can't go that way."
    if dir == 'n':
      if self.loc.north == None:
        return(error)
      self.loc = self.loc.north
    elif dir == 'e':
      if self.loc.east == None:
        return(error)
      self.loc = self.loc.east
    elif dir == 's':
      if self.loc.south == None:
        return(error)
      self.loc = self.loc.south
    elif dir == 'w':
      if self.loc.west == None:
        return(error)
      self.loc = self.loc.west
    return (self.look())

  def look(self) -> str:
    seen = "You are {}.".format(self.loc.description)
    return(seen)

  def carrying(self) -> str:
    if self.inventory != []:
      itemlist = 'You are carrying:'
      for i in self.inventory:
        itemlist = itemlist +' '+i.description + ","
      itemlist = itemlist.rstrip(',')
      itemlist += '.'
      return(itemlist)
    return("You aren't carrying anything.")

  def lookaround(self)-> str:
    visitems = 'You see:'
    for i in self.loc.items:
      visitems = visitems+' '+i.description+','
    visitems = visitems.rstrip(',')
    visitems += '.'
    return visitems

  def examine(self,descr:str) -> str:
    if self.inventory != []:
      for i in self.inventory:
        if descr in i.description:
          return i.extDescription
    for i in self.loc.items:
      if descr in i.description:
        return i.extDescription
    return ('There is no ' + descr + ' here.')

  def take(self,descr:str) -> str:
    for i in self.loc.items:
      if descr in i.description:
        if i.movable:
          self.inventory.append(i)
          self.loc.items.remove(i)
          return ('You pick up ' + i.description + '.' + self.carrying())
        return ("You can't take that!")
    return ('There is no '+ descr + ' here.')

  def drop(self,descr:str) -> str:
    for i in self.inventory:
      if descr in i.description:
          self.loc.items.append(i)
          self.inventory.remove(i)
          return ('You drop ' + i.description + '.' + self.carrying())
    return ('You are not carrying '+ descr + '.')

  def use(self,thing:str,target:str)-> str:
    thingitem = None
    targetitem = None
    useDictionary = {
        'silver fork plate of food': 'Very tasty; maybe just a little too filling. You should have shared with the d
og.',
        'a golden key a very sturdy safe': 'you opened the safe!', 'plate of food a cute puppy': 'You made
 a new best friend! The puppy is very happy.' }

    for i in self.inventory:
        if thing in i.description:
            thingitem = i
    for j in self.inventory:
        if target in j.description:
            targetitem = j
    for j in self.loc.items:
        if target in j.description:
            targetitem = j
    if thingitem == None:
        return('you are not carrying ' + thing)
    if targetitem == None:
        return('there is no ' + target)
    key = thingitem.description +' '+ targetitem.description
    if key in useDictionary:
        return (useDictionary[key])
    return("You can't use " + thing + ' on ' + target)
```