pacmanAgents.py (original)

```
# pacmanAgents.py
# ---------------
# Licensing Information: Please do not distribute or publish solutions to this
# project. You are free to use and extend these projects for educational
# purposes. The Pacman AI projects were developed at UC Berkeley, primarily by
# John DeNero (denero@cs.berkeley.edu) and Dan Klein (klein@cs.berkeley.edu).
# For more info, see http://inst.eecs.berkeley.edu/~cs188/sp09/pacman.html

from pacman import Directions
from game import Agent
import random
import game
import util

class LeftTurnAgent(game.Agent):
  "An agent that turns left at every opportunity"

  def getAction(self, state):
    legal = state.getLegalPacmanActions()
    current = state.getPacmanState().configuration.direction
    if current == Directions.STOP: current = Directions.NORTH
    left = Directions.LEFT[current]
    if left in legal: return left
    if current in legal: return current
    if Directions.RIGHT[current] in legal: return Directions.RIGHT[current]
    if Directions.LEFT[left] in legal: return Directions.LEFT[left]
    return Directions.STOP

class GreedyAgent(Agent):
  def __init__(self, evalFn="scoreEvaluation"):
    self.evaluationFunction = util.lookup(evalFn, globals())
    assert self.evaluationFunction != None

  def getAction(self, state):
    # Generate candidate actions
    legal = state.getLegalPacmanActions()
    if Directions.STOP in legal: legal.remove(Directions.STOP)

    successors = [(state.generateSuccessor(0, action), action) for action in legal]
    scored = [(self.evaluationFunction(state), action) for state, action in
successors]
    bestScore = max(scored)[0]
    bestActions = [pair[1] for pair in scored if pair[0] == bestScore]
    return random.choice(bestActions)

def scoreEvaluation(state):
  return state.getScore()
```