```
import random

################################################################
#
# three basic players
#
################################################################

def nim_minimal(n):
    return 1            # 1 is always a legal move,
                        # unless we already lost

def nim(n):
    # note: n is guaranteed to be at least 1
    return random.choice(range(1, min(n,3)+1))

def nim_best(n):
    taken = n % 4
    if taken:
        return taken
    else:                       # taken is 0, we lose - just take randomly
        # pick randomly 1 or more
        # but never more than either limit of sticks, 3
        # or available sticks, n
        return random.choice(range(1, min(n,3)+1))

def nim_human(n):
    while True: # get input until it's legal
        taken = int(input("There are %d sticks. How many do you take? (1/2/3) " %
            n))
        if taken in range(1, min(n,3)+1):
            return taken
        print("Illegal move.")

################################################################
#
# player candidates
#
################################################################

# these are the functions
player_pool = [nim_minimal, nim, nim_best, nim_human]

# transform into a dictionary with function name mapping to
# function
player_pool = { p.__name__:p for p in player_pool }

def select_players():
    players = []                # we need two

    # select the players
    while len(players) < 2:
        # select more players
        print("These are the players: %s" % "/".join(player_pool.keys()))
        p = input("Name one: ")
        if p not in player_pool.keys():
            print("Not a valid player. Select again: ")
            continue
        players.append(p)
```

```python
    print("Player %s begins, player %s plays second." % tuple(players))
    return players

def game():
    while True:
        n = int(input("Heap size? "))
        if n > 0: break     # accept only positive

    current, other = tuple(select_players()) # tuple with two elements

    # game runs

    while n > 0:            # as long as there are sticks in the heap
        print("Heap has %d sticks." % n)
        taken = player_pool[current](n) # carry out move; guaranteed legal
        print("%s takes %d sticks.\n" % (current, taken))

        n -= taken          # update heap
        current, other = other, current # now it's the other player's turn

    print("%s has lost." % current)
```

Listing 1: Solutions/nim.py