Code Walkthrough – Beeper Spreader Karel

Hi everyone! If you're working through this Karel program and need clarity, let's walk through the code **line by line** so you understand exactly what's going on.

The Program Setup

- `from karel.stanfordkarel import *`
  Loads all the commands needed to control Karel, like getting Karel's toolbox ready.

The Main Function

```
def main():
    move()
    spread_beepers()
```

- `def main():`
  The entry point - this is where the program begins.

- `move()`
  Moves Karel from its starting position **(1, 1)** to **(1, 2)**, still facing East. This sets Karel up next to the initial beeper pile.

- `spread_beepers()`
  Starts the process of distributing the beepers across the row.

The Spread Beepers Function

```
def spread_beepers():
    while beepers_present():
        pick_beeper()
        if beepers_present():
            move_to_empty_spot()
            put_beeper()
            return_to_pile()
    put_beeper()
    go_home()
```

This function handles the main task — **spreading a pile of beepers out across the row**:

- **`while beepers_present():`**
  Loop continues as long as there's a beeper at the current location (the pile).

- **`pick_beeper()`**
  Karel picks one beeper from the pile.

- **`if beepers_present():`**
  We check again: is there still more in the pile?
  If yes:

  - **`move_to_empty_spot()`**: Karel walks to the next open spot.

  - **`put_beeper()`**: Karel drops a beeper in that spot.

  - **`return_to_pile()`**: Karel goes back to the original pile to pick another.

- Once there's only **one beeper left**, Karel skips the spreading and just **places it at the current location**.

- Finally, **`go_home()`** returns Karel to the bottom-left corner - the starting point.

The Move to Empty Spot Function

```
def move_to_empty_spot():
    while beepers_present():
        move()
```

Karel walks forward **until it finds an empty cell** - one without a beeper.
This ensures that each new beeper is placed one cell farther than the last.

The Return to Pile Function

```
def return_to_pile():
    turn_around()
    move_to_the_wall()
    turn_around()
    move()
```

- Karel needs to return from the new position (e.g., 1,3) back to the beeper pile (1,2).
- `turn_around()`: Faces Karel West.
- `move_to_the_wall()`: Walks left to the wall at column 1.
- `turn_around()` again: Faces East.
- `move()`: Lands on the original beeper pile.

The Go Home Function

```
def go_home():
    turn_around()
    move()
    turn_around()
```

- This sends Karel back to the corner (1,1) when all beepers are spread.
- Uses the same trick of turning around, moving, then turning back.

The Turn Around Function

```
def turn_around():
    turn_left()
    turn_left()
```

- Karel doesn't know how to turn right directly — but turning left twice makes it face the opposite direction.

The Move to the Wall Function

```
def move_to_the_wall():
    while front_is_clear():
        move()
```

- Karel keeps walking forward until it hits a wall.
- Used for resetting position during the return trip.

Program Execution

```
if __name__ == '__main__':
    main()
```

- This ensures the program starts running from the `main()` function.
- Like pressing the **"ON" switch** for Karel's journey.

Walking Through an Example

Imagine Karel starts at position **(1, 1)** facing East, and there's a pile of 4 beepers at **(1, 2)**.

1. **Karel moves** to (1,2), stands on the pile.
2. Enters the `spread_beepers()` loop:
    - Picks one beeper.
    - Walks right to (1,3), drops it.
    - Returns to (1,2).
    - Repeats for next beeper: placed at (1,4), and so on.
3. When only one beeper is left:
    - It's placed at (1,2) without moving.
4. Karel walks back home to (1,1), facing East.