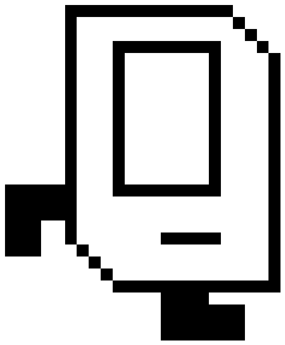
CPSC 231 ASSIGNMENT #1

Karel the Robot



In your first assignment for CPSC 231, you will learn to program Karel the Robot to perform different tasks and to solve problems within his world. This will allow you to learn the fundamentals of programming and algorithmic problem solving in a simple, visual programming environment without getting bogged down with the details of a programming language.

The first part of the assignment must be done individually, but you have the option of working with a partner for the second part. Although the deadlines are spaced evenly, the individual component is a significantly smaller, and meant to serve as a "warm-up". We intend for you to spend most of your time on the paired component, which carries twice as much weight toward your assignment grade as the individual component. If you want to work with a partner, it's a good idea to try to find one quickly and get started on the second part together as soon as you can (ideally before the first deadline).

Due Dates

Individual component	Friday, September 14, 11:59 PM*
Paired component	Friday, September 21, 11:59 PM

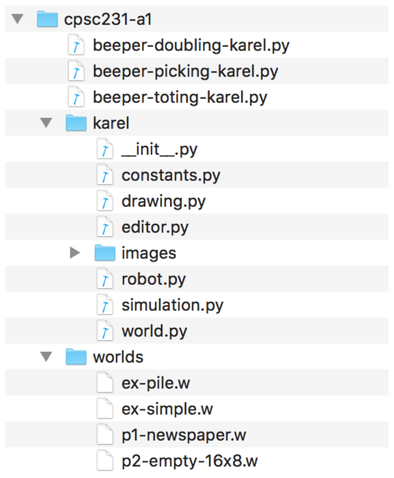
* The individual component deadline for this first assignment is a recommended deadline. This may be your first ever university assignment, so we won't penalize you for taking a little longer to get your act together. However, you must submit the individual component by the time the paired component is due.

Individual Component

We're asking you to complete the first two problems in this assignment individually. That means you must write all the code to solve these problems yourself, although you may discuss general concepts and ideas with your friends and classmates. Please refer to the course information handout or the course web site for details on the types of collaboration permitted. Don't hesitate to ask your instructors for help if you get stuck.

Before starting this assignment, you will need to install a Python 3 programming environment and the Booksite Library modules provided to accompany our course text, *Introduction to Programming in Python*. Detailed instructions on how to do this can be found on the booksite, `https://introcs.cs.princeton.edu/python/home/`. If you are working on one of our CPSC lab machines, you can start at the "Downloading and Installing the Booksite Library" step. You might rightfully be wondering why you need to set up our full Python environment to program Karel. Don't worry about that for now, as it will all become clear within a few weeks.

When you've verified that your Python programming environment works, download the `karel` module from the course assignments page and unzip it into the folder on your computer where you will be doing this assignment. If your folder structure look something like what is shown on the right, you should be ready to go! You can run one of the provided example Karel programs to see Karel in action by running a command like this:



```
cpsc231-a1
    beeper-doubling-karel.py
    beeper-picking-karel.py
    beeper-toting-karel.py
    karel
        __init__.py
        constants.py
        drawing.py
        editor.py
        images
        robot.py
        simulation.py
        world.py
    worlds
        ex-pile.w
        ex-simple.w
        p1-newspaper.w
        p2-empty-16x8.w
```

```
python3 beeper-toting-karel.py worlds/ex-simple.w
```

This command will use your Python 3 environment to run the Karel program contained in the file `beeper-toting-karel.py` on the world described in the file `ex-simple.w` from the `worlds` folder.

Problem 1: Picking up the Paper

Karel is a little old-fashioned,[1] and still likes to have his newspaper delivered so that he can read it in print. Every morning, Karel is awakened in bed when his newspaper, represented by a beeper, is thrown on the front porch of his house. Program Karel to retrieve the paper and bring it back to bed. The initial situation is exactly as shown in Figure 1 (and encoded in the world file `p1-newspaper.w`). The final situation must have Karel back in bed (same location and facing the same direction) with the newspaper.

[1] Believe it or not, Karel was invented in the year 1981! Considering how quickly computer technology evolves, we can truly call Karel a timeless instructor of computer science.

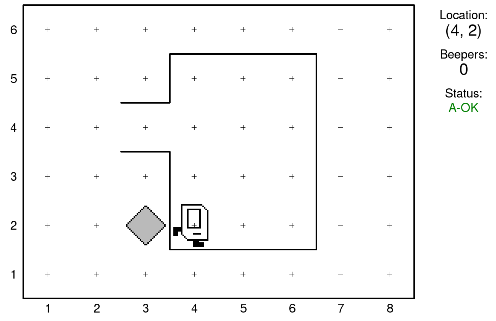


Figure 1: Karel wakes up when the newspaper is thrown onto the porch in front of his house.

Problem 2: Writing with Karel

Now that you've met Karel, it would be nice to introduce yourself to Karel as well. You can tell Karel your name, but since he has no memory, he cannot remember it. He can write it down though!

Program Karel to write your initials in his world using beepers. In this problem, Karel will start in an empty world on the corner of

1st avenue and 1st street, facing east. The final situation may look like that shown in Figure 2, but with your own initials (2, 3, or more letters) appearing instead. You may use any "typeface" you like, as long as it is clearly legible. You may ask Karel to write your full name if you prefer, though it would likely take up too much space. The default world for this problem measures 25 avenues across by 10 streets high (`p2-empty-25x10.w`), but you can adjust the size if you want (up to the 50×50 maximum).

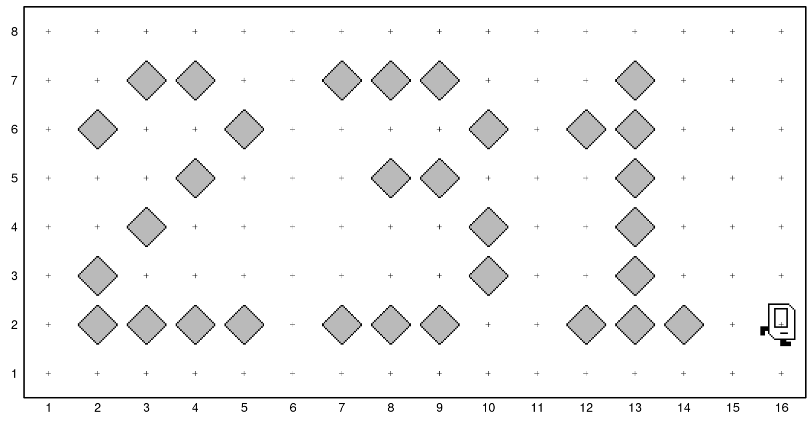


Figure 2: Karel's world after he has written the numbers 231 using beepers.

Paired Component

We encourage you to work with a partner to complete the following problems. You may choose your own partner, but remember that you must work with a *different partner* for each assignment in this class! If you are solving these problems as a pair, one submission is sufficient for both students.

Remember that you may only use use valid Karel the Robot instructions, program syntax, and constructs that you learned in class (*i.e.* those that appear on the *Karel Reference Card*) when solving these problems. If you've done some programming before, and especially if you already know the Python language, you may be tempted to introduce additional programming constructs such as variables, parameters, return values, and the like. Don't do this, because Karel has no idea what they mean! If you do use constructs that Karel does not understand, your program may still work, but you will not obtain full credit for solving the problem.

The problems in this component can be surprisingly difficult, and are designed to exercise the problem-solving and programming strategies you learned in class. It may be helpful to keep the following in mind when working on these problems.

- Decompose the problem into smaller and more manageable pieces. Test each new instruction individually to ensure that it works as intended.
- Name your new instructions clearly and add comments to your code to help both yourself and your partner understand their function.
- Karel worlds (`.w` files) are just human-readable text files. You can edit them with any text editor to create additional scenarios to run your programs on.
- Your program for each problem should work in a variety of different worlds that match the problem specifications. Before you submit your programs, test them out in as many different worlds as you can.

Problem 3: Karel the Gardener

Karel has taken a part-time job as a gardener. Karel's specialty is planting beepers. Karel's current ask is to plant one, and only one, beeper on each street corner along a cross-shaped wall arrangement like that shown in Figure 3.

Karel starts facing east in his world, on the corner of the first street south of the southernmost wall and the first avenue west of the westernmost wall, with enough beepers to plant the full arrangement. The cross-shaped wall arrangement is always symmetric, but can be of any size. There will always be at least one open street or avenue around the edge so that Karel can walk around the walls. Karel's final location and direction do not matter.

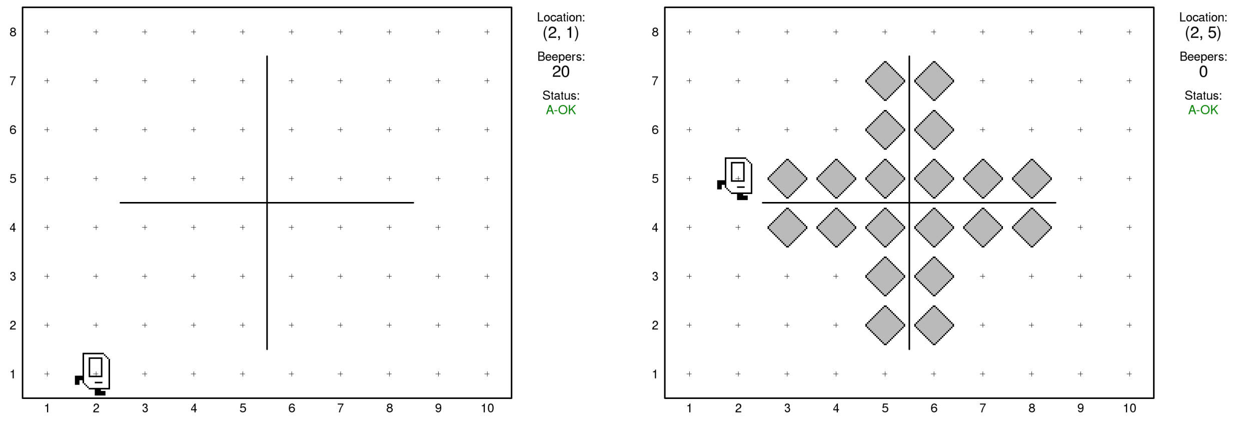


Figure 3: The initial cross-shaped wall arrangement is shown on the left. The final garden after Karel has finished his planting is shown on the right.

Your final program for this problem must be able to plant a cross-shaped garden of any size, provided the initial world conforms to the above specifications. However, one strategy you can use to solve this problem is to write a program that plants a garden exactly of the size depicted above (`p3-garden3.w`) first. Try to use the same code sequence, by means of defining a new instruction, to plant each "quadrant" of the garden. Then see if you can add conditional statements to your program to make it work for cross-shaped gardens of arbitrary size.

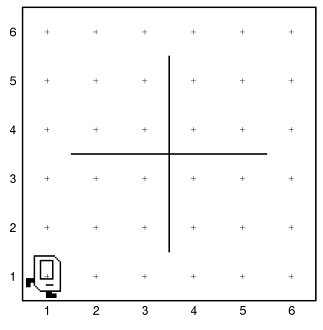


Figure 4: Another garden that Karel should be able to plant by running the same program for this problem.

Problem 4: Midpoint Karel

Karel starts in an empty world in this problem, and your job is to program Karel to perform a simple task: place a single beeper halfway along the length of first street. For example, if Karel lives in 5×5 world, the initial and final configurations should look as shown in Figure 5.

Note that the final configuration should have only a single beeper at the centre of 1st street, with Karel standing on top of it. In your program, Karel is allowed to place additional beepers wherever he wants to, but must pick them all up again before your program terminates.

Karel's initial situation conforms to the following specifications:

- Karel starts on the corner of 1st street and 1st avenue, facing east,

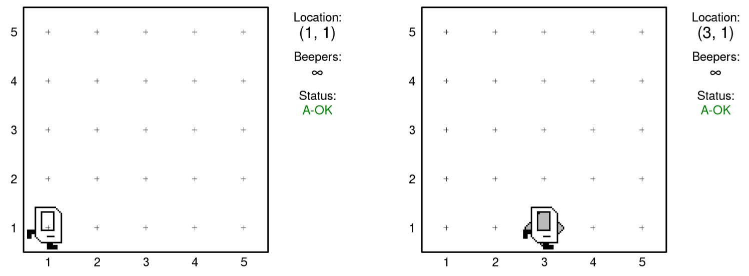


Figure 5: Starting in an empty world as shown on the left, Karel must find the midpoint of 1st street, place a single beeper there, then stop on top of it.

with an infinite number of beepers in his bag.

- The world has no interior walls and initially has no beepers.
- The world need not be square, but has at least as many streets as it does avenues.

If the width of the world is odd, Karel must put the beeper exactly in the middle intersection on 1st street. If the width is even, Karel may drop the beeper on either of the two central intersections. It does not matter which direction Karel is facing at the end.

There are many different approaches you can take to solve this problem. The interesting part of this exercise is to come up with a strategy that works. Can you think of more than one?

Problem 5: Checkerboard Karel

Your final job for this assignment is to program Karel to create a checkerboard pattern of beepers inside an empty rectangular world. For example, on a standard 8×8 checkerboard, the initial and final situations are shown below. Karel starts on the corner of 1st street and 1st avenue facing east, and his final location and direction do not matter. Note that the pattern always has a beeper on the southwestern-most corner.

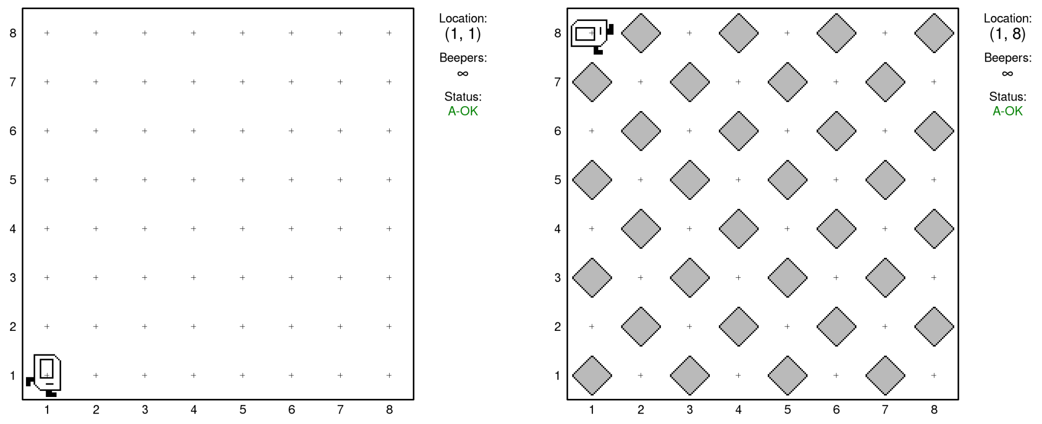


Figure 6: The initial and final state of a standard 8×8 checkerboard.

As you think about how to solve this problem, keep in mind that your solution should work with different-sized rectangular worlds.

Odd-sized checkerboards can be tricky. Other special cases to consider are worlds with only one street or avenue. Some examples of oddly-shaped worlds are provided for you, but understand that your TA may evaluate your program on worlds of any size!

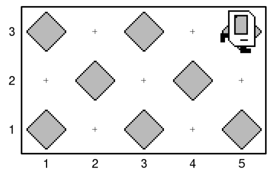


Figure 7: Some worlds can be quite challenging! Ensure your program can produce this pattern in a 5×3 world.

Bonus Problem: Hungry Hungry Karel

After working so hard to help you complete this assignment, Karel has become very, very hungry. Luckily for Karel, we're in the middle of Stampede week when free pancake breakfasts abound. What better way to satisfy your hunger?

At one such breakfast, Karel is standing in front of a row of pancake stacks, represented by piles of beepers of course. He has noticed that not all the stacks are the same size. Being very hungry, Karel of course wants to eat the largest stack of pancakes. However, he doesn't want to appear too greedy, especially since the breakfast is free, so he decides he will eat the *second* largest stack of pancakes.

Write a program that instructs Karel to pick up all the beepers in the second-largest pile of beepers in his world. You may assume that the state of Karel's world satisfies the following properties:

- Karel starts in his favourite location, the corner of 1st avenue and 1st street, facing east.
- All piles of beepers will appear on 1st street at intersections with even-numbered avenues (*i.e.* there is an empty avenue between each pile).
- There is at least one empty avenue east of the easternmost pile.
- There are at least as many streets as there are beepers in the largest pile.
- There are at least two piles of beepers in Karel's world.
- All piles have a distinct number of beepers.

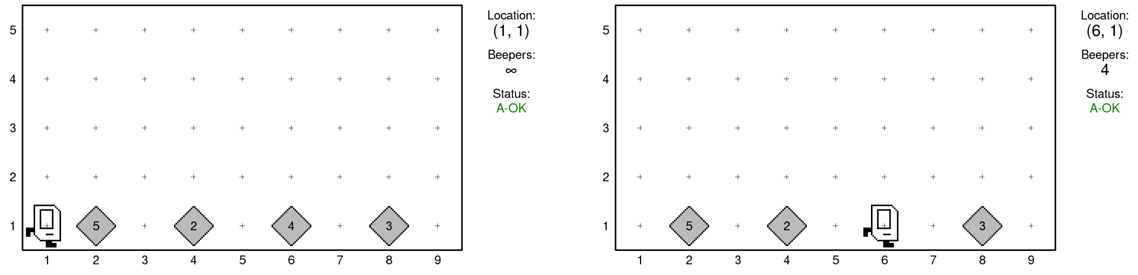


Figure 8: The initial world with stacks of pancakes is shown on the left. The final state of the world, after Karel has eaten the second-largest stack, is shown on the right.

After running your program, Karel should have picked up all the beepers in the second-largest pile. While executing your program, Karel may inspect and rearrange the other piles any way he'd like, but before your program ends, he must put all those beepers back exactly as they were, so that nobody would suspect they've ever been touched. Karel's final location and direction do not matter.

Submission

After you have completed this assignment, you should have written a total of five Karel the Robot programs (or six, if you completed the bonus) saved as `.py` files. Please ensure that your files are named descriptively, with the problem number included, so that your TA can easily see which program is associated with each problem.

Use the University of Calgary Desire2Learn system[2] to submit your assignment work online. Log in using your UofC eID and password, then find our course, CPSC 231, in the list. Then navigate to Assessments → Dropbox Folders, and find the folder for Assignment #1 here. Upload your programs for problems 1 and 2 in the individual folder, and your other programs in the paired folder (if your partner hasn't already done so). We only need one submission for each pair of students, but don't forget to write both of your names on the submission! If you make a mistake or need to update your files, don't worry, just create another submission. We will only evaluate your latest submission.

[2] `http://d2l.ucalgary.ca`

If you are using one or more of your grace "late days" for this assignment, indicate how many you used in the note accompanying your submission. Remember that late days will be counted against *both* partners in the paired component. If you completed the bonus problem, please indicate that here as well.

Credits

Problems 1, 2, and 3 in this assignment were either inspired by or drawn from *Karel the Robot: A Gentle Introduction to the Art of Programming* by Richard Pattis,[3] who originally came up with the concept of Karel the Robot. Problems 4 and 5 were drawn from the CS 106A course taught at Stanford University. To the best of our knowledge, these were originally authored by Eric Roberts.

[3] Richard E. Pattis. *Karel the Robot: A Gentle Introduction to the Art of Programming*. John Wiley & Sons, 2nd edition, 1995