Problems 3 and 4: Picobot

courses.edx.org/courses/course-v1:HarveyMuddX+CS005x+2T2016/courseware/28a22dfe1aae4c939c1072aac3466c2c/c870bfc9bf24441d9313b416c09c0865/

Favoris

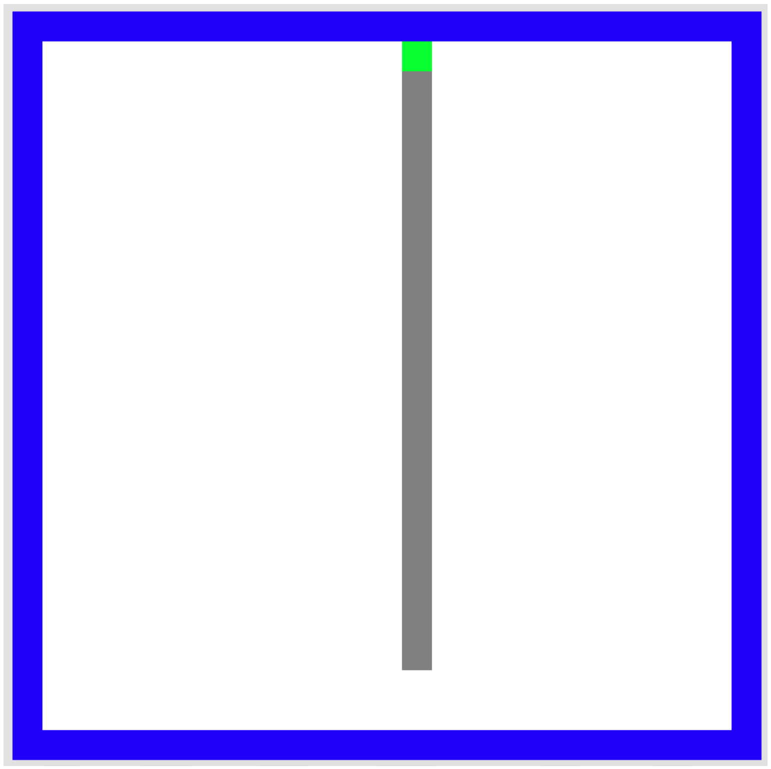
Week 1: Introduction to Computation > Homework 1 > Problems 3 and 4: Picobot

When you're finished with this assignment, submit your code at the bottom of this page.

This problem explores a simple robot, named "Picobot," whose goal is to completely traverse its environment.

Here is a link to the Picobot page. You should open Picobot in a new page and keep that page open while you work on this assignment.

Picobot starts at a random location in a room—you don't have control over Picobot's initial location. The walls of the room are blue; picobot is green, and the empty area is white. Each time Picobot takes a step, it leaves a grey trail behind it. When Picobot has completely explored its environment, it stops automatically.



Surroundings

Not surprisingly, Picobot has limited sensing power. It can only sense its surroundings immediately to the north, east, west, and south of it.

These surroundings are represented using four letters in North, East, West, South (NEWS) order.

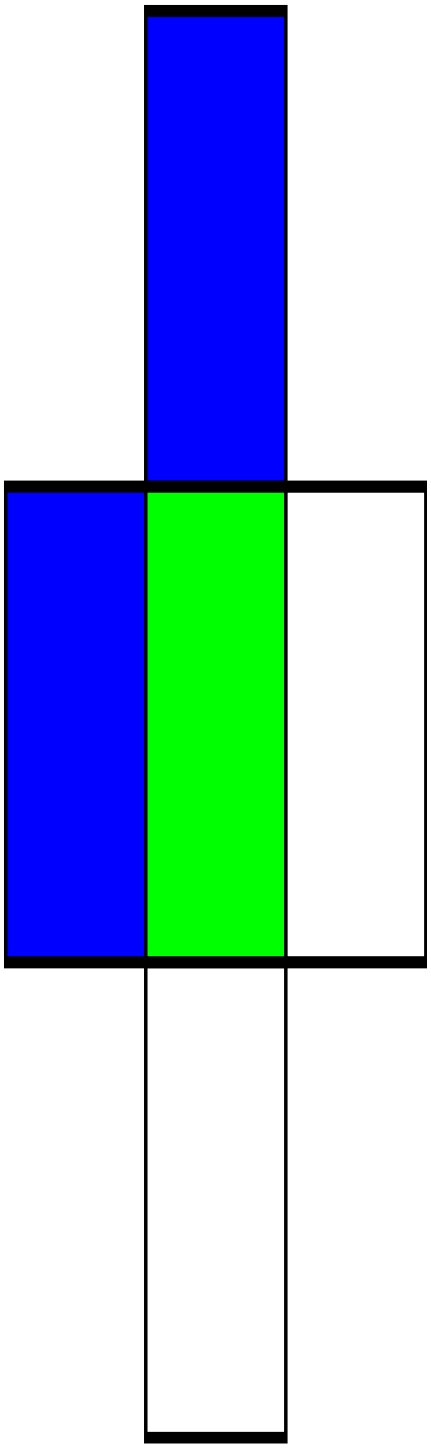
For example, if Picobot is in the upper left corner of the room, it has these surroundings:

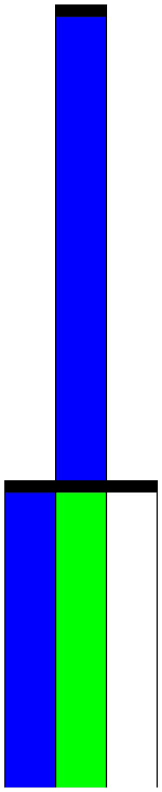
In the above image, Picobot sees a wall to the north and west and it sees nothing to the east or south. This set of surroundings is represented as `NxWx`.

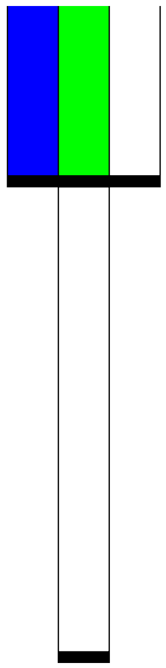
The four squares surrounding picobot are always listed in NEWS order. An `x` represents empty space, and the appropriate direction letter (`N`, `E`, `W`, or `S`) represents a wall blocking that direction.

Here are all of the possible picobot surroundings:

- `xxxx`
- `Nxxx`
- `xExx`
- `xxWx`
- `xxxS`
- `NExx`
- `NxWx`







NxxS
xEWx
xExS
xxWS
NEWx
NExS
NxWS
xEWS
NEWS

State

Picobot's memory is also limited. In fact, it has only a single value from 0 to 99 available to it. This number is called picobot's **state**. In general, "state" refers to the relevant context in which computation takes place. Here, you might think of each "state" as one piece—or behavior—that the robot uses to achieve its overall goal.

Picobot always begins in state 0.

The state and the surroundings are all the information that picobot has available to make its decisions!

Rules

Picobot moves according to a set of rules of the form

```
StateNow   Surroundings   ->   MoveDirection
NewState
```

For example,

```
0   xxxS   ->   N
0
```

is a rule that says "If Picobot starts in state `0` and sees the surroundings `xxxS`, it should move `N`orth and stay in state `0`."

The `MoveDirection` can be `N`, `E`, `W`, `S`, or `X`, representing the direction to move or, in the case of `X`, the choice not to move at all.

If this were Picobot's only rule and if picobot began (in state `0`) at the bottom of an empty room, it would move up (north) one square and stay in state `0`. However, **Picobot would not move any further**, because its surroundings would have changed to `xxxx`, which does not match the rule above.

Wildcards

The asterisk `*` can be used inside surroundings to mean "I don't care whether there is a wall or not in that position." For example, `xE**` means "There is no wall North, there is a wall to the East, and there may or may not be a wall to the West or South."

As an example, the rule

```
0   x***   ->   N
0
```

is a rule that says "If Picobot starts in state `0` and sees **any surroundings without a wall to the North**, it should move North and stay in state `0`."

If this new version (with wildcard asterisks) were Picobot's only rule and if Picobot began (in state `0`) at the bottom of an empty room, it would first see surroundings `xxxS`. These match the above rule, so Picobot would move North and stay in state `0`. Then, its surroundings would be `xxxx`. These also match the above rule, so Picobot would again move North and stay in state `0`. In fact, this process would continue until it hit the "top" of the room, when the surroundings `Nxxx` no longer match the above rule.

Comments

Anything after the pound sign (`#`) on a line is a comment (as in Python). Comments are human-readable explanations of what is going on. They are ignored by Picobot. Blank lines are ignored as well.

An Example

Consider the following set of rules:

```
# state 0 goes N as far as possible
0 x*** -> N 0   # if there's nothing to the N, go
N
0 N*** -> X 1   # if N is blocked, switch to state
1

# state 1 goes S as far as possible
1 ***x -> S 1   # if there's nothing to the S, go
S
1 ***S -> X 0   # otherwise, switch to state 0
```

Recall that Picobot always starts in state `0`. Picobot now consults the rules from **top to bottom** until it finds the first rule that applies. It uses that rule to make its move and enter its next state. **It then starts all over again** , looking at the rules and finding the first one from the top that applies.

In this case, Picobot will follow the first rule up to the "top" of its environment, moving north and staying in state `0` the whole time. Eventually, it encounters a wall to its north. At this point, the topmost rule no longer applies. However, the next rule `0 N*** -> X 1` does apply now! So, Picobot uses this rule which causes it to stay put (due to the `X`) and **switch to state** `1`. Now that it is in state `1`, neither of the first two rules will apply. Picobot follows state `1`'s rules, which guide it back to the "bottom" of its environment. When Picobot gets to the bottom, the second rule in state `1` will apply, and it will stop and return to state `0`. So, these rules will make Picobot go up and down in the same column again and again.

Your Assignment

For this assignment, your task is to design two different sets of Picobot rules. Both sets of rules must work from **any starting position** in the room. Remember to click on the "Enter rules for Picobot" button before you try to run Picobot!

The Picobot site will not save your work. Make sure that you save a copy of your work before closing your Picobot window.

Homework 1, Problem 3

Design one set of rules that will allow Picobot to completely cover an empty square room.

Homework 1, Problem 4

Design a second set of rules that will allow Picobot to completely cover a maze. Click on the `-->` button next to the word MAP below the Picobot room to get to the maze map from the empty room map.

If you want to spend more time with Picobot after you've finished the required problems, take a look at the Extra

Challenges on the last page of this homework!

Submit Homework 1, Problem 3

25.0/25.0 points (graded)

To submit your Homework 1, Problem 3 code, copy it from the Picobot window and paste it into the box below. After you've pasted your code below, click the "Check" button.

Remember, this is the **empty room**!

IMPORTANT: Make sure that there aren't spaces at the beginning of your code, and that you copied all of the characters. If there are extra spaces or you are missing spaces, our server won't be able to run your code and we won't be able to give you any of the points you deserve for your hard work.

```
# paste your code here
0 x*** -> N 0
0 N*x* -> W 1
0 N*W* -> E 2

1 ***x -> S 1
1 **xS -> W 0
```

```
1 **WS -> E
3

2 ***x -> S
2

2 ***S -> E
3

3 x*** -> N
3

3 N*** -> E
2
```

Press ESC then TAB or click outside of the code editor to exit
correct

correct

Test results
CORRECT See full outputSee full output

You have used 1 of 3 attempts Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Submit Homework 1, Problem 4

25.0/25.0 points (graded)

To submit your Homework 1, Problem 4 code, copy it from the Picobot window and paste it into the box below. After you've pasted your code below, click the "Check" button.

Remember, this is the **maze**!

IMPORTANT: Make sure that there aren't spaces at the beginning of your code, and that you copied all of the characters. If there are extra spaces or you are missing spaces, our server won't be able to run your code and we won't be able to give you any of the points you deserve for your hard work.

```
0 xE** -> N
0

0 *x** -> E
1

0 NEx* -> W
2

0 NEW* -> S
3



1 xE*S -> N
0

1 *x*S -> E
1
```

```
1 NE*S -> W
2
```

9

```
1 ***x -> S
3
```

10

11

```
2 x*** -> N
0
```

12

```
2 N*WS -> E
1
```

13

```
2 N*x* -> W
2
```

14

```
2 N*Wx -> S
3
```

15

16

```
3 *EWS -> N
0
```

17

```
3 *xWS -> E
1
```

18

```
3 **x* -> W
2
```

19

```
3 **Wx -> S
3
```

Press ESC then TAB or click outside of the code editor to exit
correct

correct

Test results
CORRECT See full outputSee full output

You have used 1 of 3 attempts Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.