CS 97: Practical and Practices of Computing

Lun Liu

Welcome to CS97 Discussion!

- Course website: https://ccle.ucla.edu/course/view/17F-COMSCI97-1
- Piazza: https://piazza.com/ucla/fall2018/cs97
  - Post your questions here
  - If your question contains your code, post a private note so only instructors can see!
- Discussion session materials: https://github.com/pakkaliu/CS97-Fall18
  - Click on “week1” to access the materials for the first week
  - Practice problems will be posted before discussion
  - Slides (sample solutions) will be posted after discussion

Welcome to CS97 Discussion!

- Lun Liu: lunliu93@cs.ucla.edu
- Office Hours: BH 2432, Wednesdays 5 - 7pm
  - All CS office hours are held here, so look around
  - For help, not answers
- This section
  - Practice programming with new problems
  - Review main concepts from class
  - Smaller setting for questions
  - Get help from LAs: Matt, Anirudh!

If-elif-else review

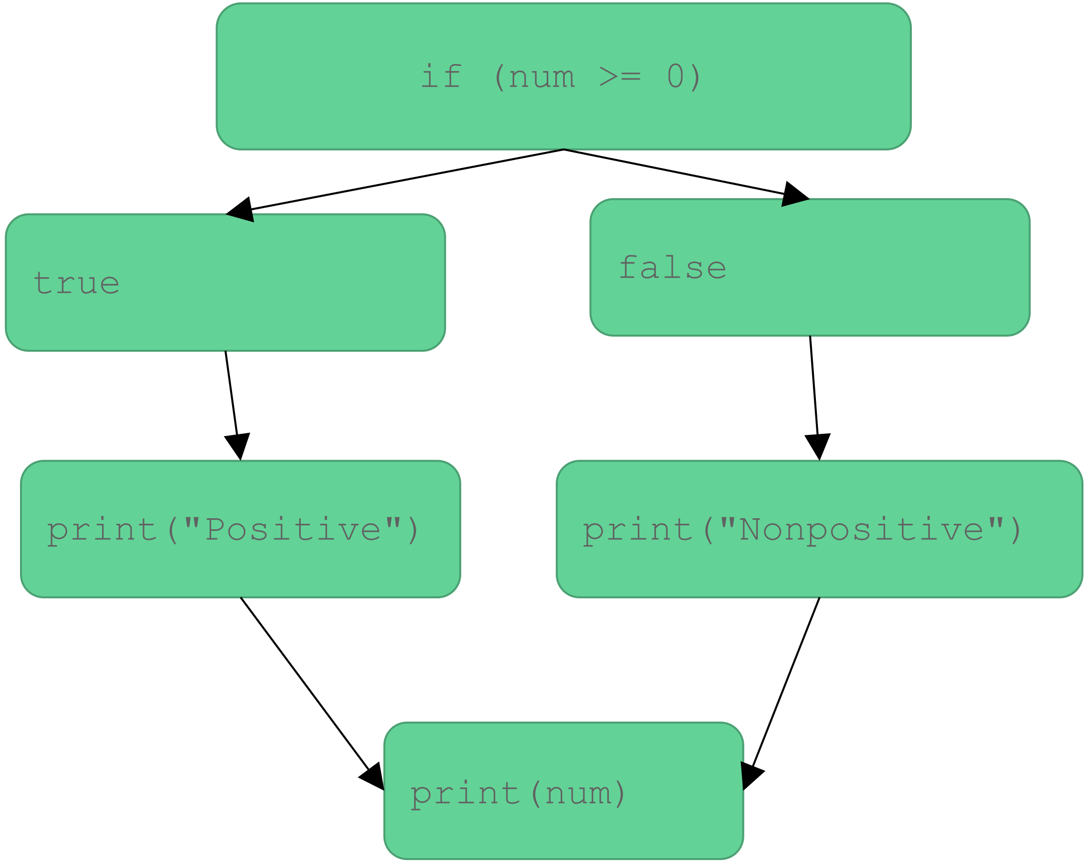
```
if expression1:
   statement(s)
elif expression2:
   statement(s)
elif expression3:
   statement(s)
else:
   statement(s)
```

Remember: if-elif-else implies that each "case" is *exclusive.*

What would the following print out if x were 5?
What if line 3 were changed to a regular if?

```
if x < 10:
    print("I'm under ten!")
elif x == 5:
    print("I'm five!")

else:
    print("I give up")
```

```
def isPositive(num):
    if num >= 0:
        print("Positive: ")
    else:
        print("Nonpositive: ")
    print(num)
```



Flowcharting can be a great problem solving tool if you're stuck!!

Common Mistakes-Week 1 Edition

1) "Strings" versus variables. Remember that quotes gives you literally exactly what is between the quotes; that's why they're called string *literals*. Ex:
```
>> num = 8
>> print("num")
```
2) Equality versus assignment, aka one versus two equal signs. Ex:
```
>> if x = 7:  # Runs like it's right, but it is not comparing x with 7
```
or
```
>> answer == "positive"
```
3) "I changed something but it still has the old behavior!!!"
You need to re-run the module.

Practice Time!

- Go to https://github.com/pakkaliu/CS97-Fall18/tree/master/week1 and try to solve those Python problems
- Discuss with others if you want, or ask our LAs

If-else practice: #0

Write a function that returns the absolute value of a number

```
def abs(x):
    if x > 0:
        return x
    else:
        return -x
```

If-else practice: #1

You've been asked to write the customer interface for ticket pricing at a theme park! Children two years old and under can enter the park for free. Children 12 and under get a discounted price of $40. Seniors also get a discounted price; anyone 65 or over only has to pay $60. Everyone else has to pay $85.

Write a function that takes in the age of a guest and returns the correct price.

Problem 1: (A possible) solution

```
def getPrice(age):
    if age <= 2:
        price = 0
    elif age <= 12:
        price = 40
    elif age < 65:
        price = 85
    else:
        price = 60
    return price
```

"There are two hard things in computer science: cache invalidation, naming things, and off-by-one errors."

Beware the off-by-one error! Often times, programs work correctly, except for being off by ONE number; for example, `age < 2` instead of `age <=  2` makes your program wrong only if the age is 2!

. What are some ages you would plug in to your function to test that it always does what it's supposed to?

Problem 1 Revisited

The theme parks wants to offer a special promotion to adults, but only if they live in California and only if they don't already get a special price (i.e. adults between the ages of 12-65). If they do, they get 15% off the regular entrance price. Your function now takes in a second argument that represents the two letter state code("AL", "FL", "CA", "NV", etc).

How would you modify the code you wrote for the previous problem to incorporate this new policy?

Problem 1 Revisited (A possible) solution

```python
def getPromoPrice(age, state):
    if age <= 2:
        price = 0
    elif age <= 12:
        price = 40
    elif age < 65:
        price = 85
        if state == "CA":
            price = price * (1 - 0.15)
    else:
        price = 60
    return price
```

Problem #2

Write a function that takes in three numbers and returns the smallest of the three.

Food for thought: what happens if you put in three of the same number? Which lines run?

```python
def smallest(x, y, z):
        if x < y:
                if x < z:
                        return x
                else:
                        return z
        else:
                if y < z:
                        return y
                else:
                        return z
```

Bonus problem:

Write a function that tells you whether or not a point is inside a circle.

- What arguments would you need?
- The traditional mathematical solution needs the square root function, but you don't need it to solve this problem. Can you do it using what you know?
- Reminder: pow(base, exponent) gives you the power. So pow(num, 2) can square something.

No solution given for this: you can look this up and ponder it on your own time!

Picobot! Rule refresher

- Rules consist of a few parts:
  - < current state >
  - < N > < E > < W > < S >
  - The "->" transition
  - < next direction >
  - < new state >
- Some special characters:
  - "*" means "Either a wall or free space"
  - "x" before the arrow means "free space"
  - X after the arrow means "do nothing"
- Example:
  - 1 xE** -> N 2
  - In English this means: "In state 1, if Picobot senses a wall to its east and a free space to its north, then it should go north and change to state 2, no matter what is to its south or west.

Picobot practice problem!

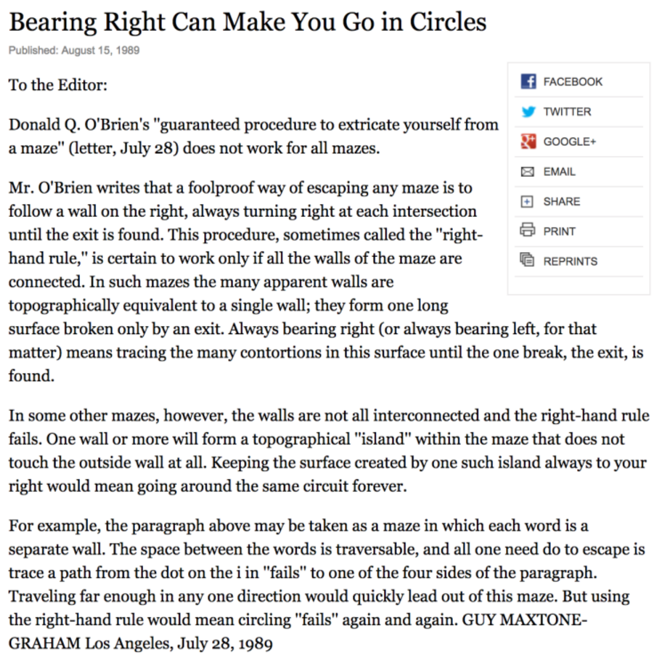
- We already made Picobot travel to the northeast corner of the board in straight lines in class.
  - 0 *x** -> E 0
    0 *E** -> N 0
- And a row, using states:
  - 0 *x** -> E 0
    0 *E** -> X 1
    1 **x* -> W 1
- How do we make Picobot travel in a zigzag pattern across the (empty) board?
  - From any given location, Picobot should take one step north, then one step east, then one step north ad infinitum until it reaches a wall and can't go any farther
  - It should never take two consecutive steps in the same direction.

Picobot practice problem!

```
# state 0 with no walls to the N: go one step N

0 x*** -> N 1



# state 1 with no walls to the E: go one step E

1 *x** -> E 0
```

Right hand rule!

It's so exciting, it's been the subject of heated debate in the NYTimes OpEd section!



Bearing Right Can Make You Go in Circles

Published: August 15, 1989

FACEBOOK
TWITTER
GOOGLE+
EMAIL
SHARE
PRINT
REPRINTS

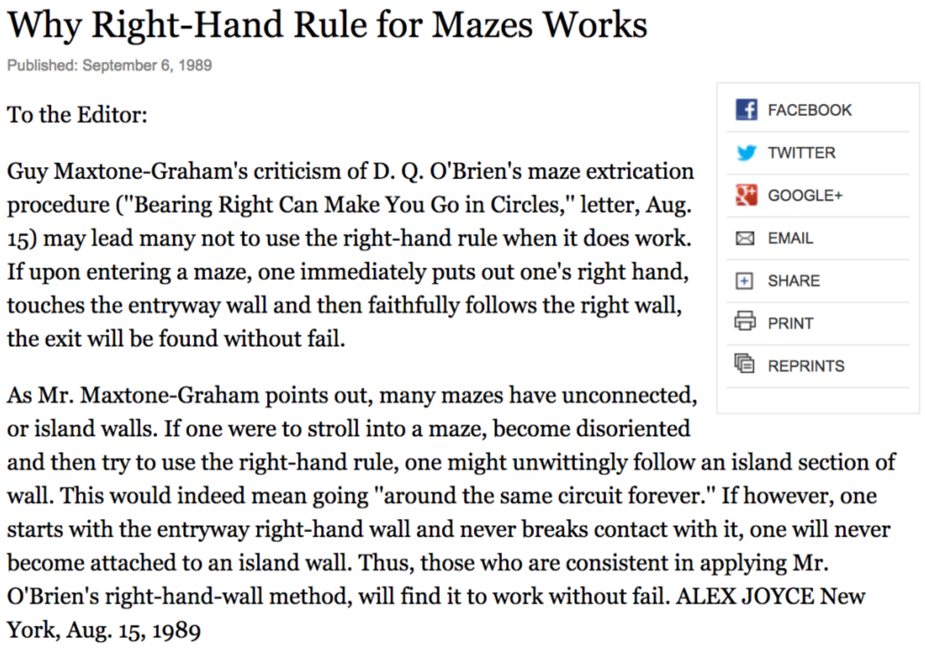
To the Editor:

Donald Q. O'Brien's "guaranteed procedure to extricate yourself from a maze" (letter, July 28) does not work for all mazes.

Mr. O'Brien writes that a foolproof way of escaping any maze is to follow a wall on the right, always turning right at each intersection until the exit is found. This procedure, sometimes called the "right-hand rule," is certain to work only if all the walls of the maze are connected. In such mazes the many apparent walls are topographically equivalent to a single wall; they form one long surface broken only by an exit. Always bearing right (or always bearing left, for that matter) means tracing the many contortions in this surface until the one break, the exit, is found.

In some other mazes, however, the walls are not all interconnected and the right-hand rule fails. One wall or more will form a topographical "island" within the maze that does not touch the outside wall at all. Keeping the surface created by one such island always to your right would mean going around the same circuit forever.

For example, the paragraph above may be taken as a maze in which each word is a separate wall. The space between the words is traversable, and all one need do to escape is trace a path from the dot on the i in "fails" to one of the four sides of the paragraph. Traveling far enough in any one direction would quickly lead out of this maze. But using the right-hand rule would mean circling "fails" again and again. GUY MAXTONE-GRAHAM Los Angeles, July 28, 1989



Why Right-Hand Rule for Mazes Works

Published: September 6, 1989

FACEBOOK
TWITTER
GOOGLE+
EMAIL
SHARE
PRINT
REPRINTS

To the Editor:

Guy Maxtone-Graham's criticism of D. Q. O'Brien's maze extrication procedure ("Bearing Right Can Make You Go in Circles," letter, Aug. 15) may lead many not to use the right-hand rule when it does work. If upon entering a maze, one immediately puts out one's right hand, touches the entryway wall and then faithfully follows the right wall, the exit will be found without fail.

As Mr. Maxtone-Graham points out, many mazes have unconnected, or island walls. If one were to stroll into a maze, become disoriented and then try to use the right-hand rule, one might unwittingly follow an island section of wall. This would indeed mean going "around the same circuit forever." If however, one starts with the entryway right-hand wall and never breaks contact with it, one will never become attached to an island wall. Thus, those who are consistent in applying Mr. O'Brien's right-hand-wall method, will find it to work without fail. ALEX JOYCE New York, Aug. 15, 1989

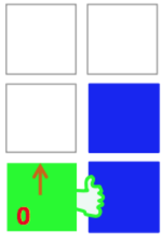
Suppose Picobot wants to traverse a maze ***with its right hand <u>always</u> on the wall...***

(A) CORRIDOR rule

If you're facing N with a wall at right and space ahead then go forward"

```
0    xE**    ->    N    0
```

state 0 means "still facing north"



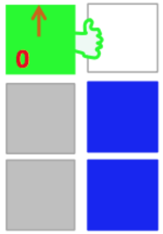
(A)

(B) INTERSECTION rule

"If you're facing North and lose the wall, then get over to the wall now!"

```
0    *x**    ->    E    1
```

state 1 means "now facing east"



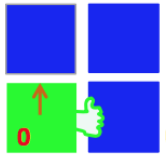
(B)

(C) DEAD END rule

Write 1 or 2 rules to tell Picobot to do the right thing if it hits a dead end.

```
0    NE**    ->    X    2
```

state 2 means "now facing west"



(C)

Repeat this IDEA for all four states, representing all four ***facing directions.***