PROGRAMMING

Lecture 08

Hanbat National University
Dept. of Computer Engineering
Changbeom Choi

OUTLINE

Adding beepers

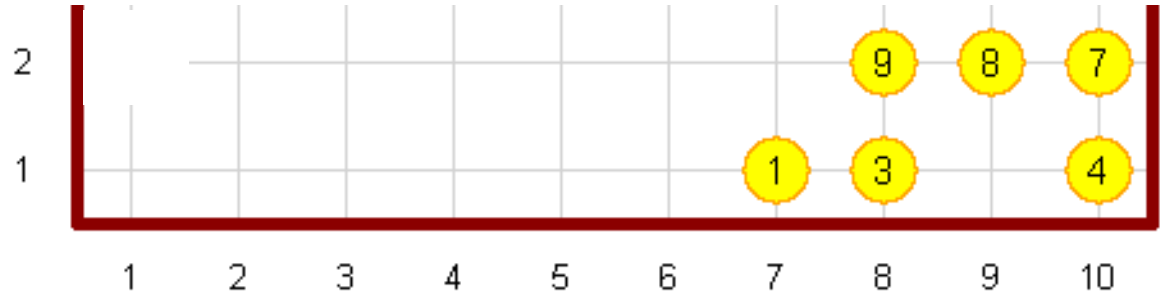
Triangular inequality

Drawing graphs

ADDING BEEPERS

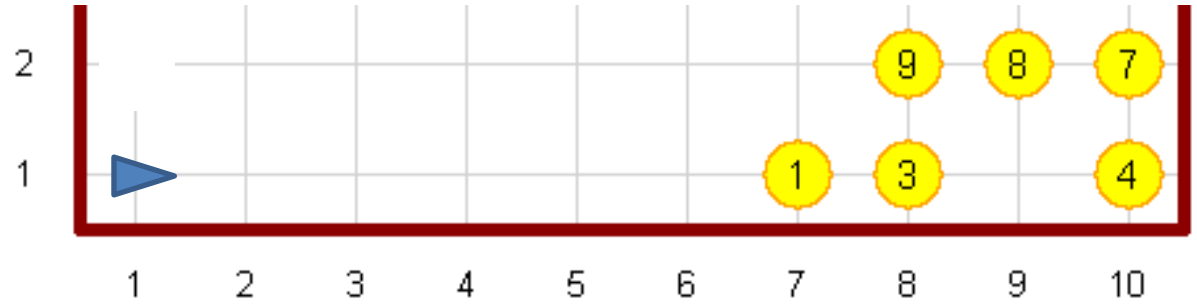
PROBLEM 1: ADDING BEEPERS

Given beepers in two rows of a 2D world, Hubo wants to **collect** the **beepers** in **each column** and **put them** at **the place in the first row**. Hubo should **move back** to the **starting position** and also **recover his orientation** after finishing his task. You may assume that beepers, if any, are initially placed in **the first two rows**, that is, the first and second rows. Not every column has beepers on two places: A column may have no places with beepers and other column may have one place with beepers as shown below:

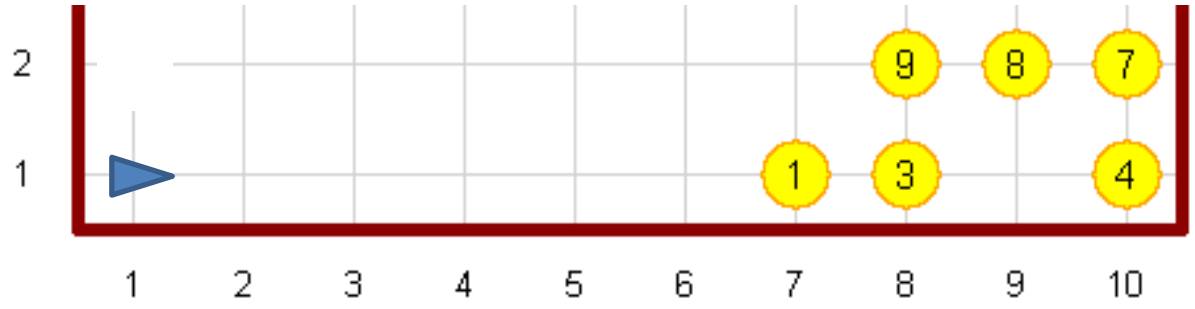


Pseudo code

1. Process all columns:
    While not blocked:
        Collect beepers in a column.
        Move to the next column.
    Collect beepers in the last column. Why?
2. Move back to the initial position.



```
Collect beepers in a column:
    Go up and pick beepers:
        Move up to the second row.
        Collect all beepers in the position if any.
    Go down and add beepers:
        Move down to the first row.
        Add all beepers.
```



How to collect all beepers in a place

```
def pick_beepers():
    while hubo.on_beeper():
        hubo.pick-beeper()
```

How to add all beepers in a place()

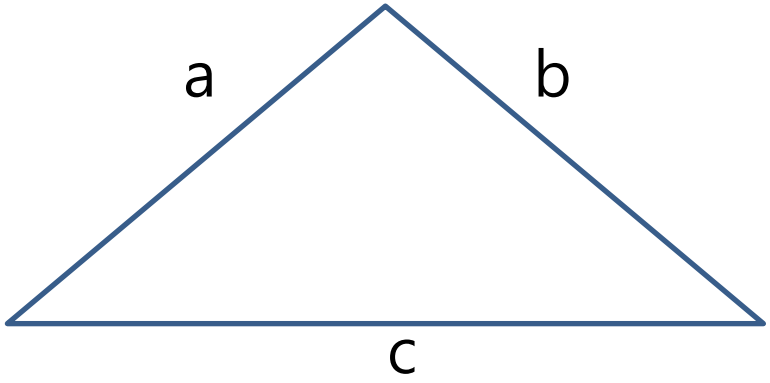
```
def add_beepers():
    while hubo.carries_beepers():
        hubo.drop_beeper()
```

TRIANGULAR INEQUALITY

PROBLEM 2: TRIANGULAR INEQUALITY

Given three numbers a, b, and c, it is possible to form a triangle whose sides have length a, b, and c if and only if the **triangle inequality** holds. In other words, every side should be shorter than the sum of the other two sides. Write a program that checks if a triangle can be formed with the three numbers. The three numbers, a, b, c are provided by the user with a built in function **input** . As the output, the three numbers together with "True" or "False" should be printed depending on that the triangular inequality holds true or not.

Triangular inequality



$a + b > c$, $b + c > a$, and $c + a > b$

Pseudo code

1. Input three numbers.
2. Check the triangular inequality for these numbers and report result.
3. Repeat Steps 1 and 2 until no more input is given.

A while-loop

Main program

```
while True:
    a, b, c = take_numbers()
    if check_inequality(a, b, c):
        print  (a, b, c, “True”)
    else:
        print (a, b, c, “False”)
    if input(“Go for a more check ?”)  != “yes”:
        break
```

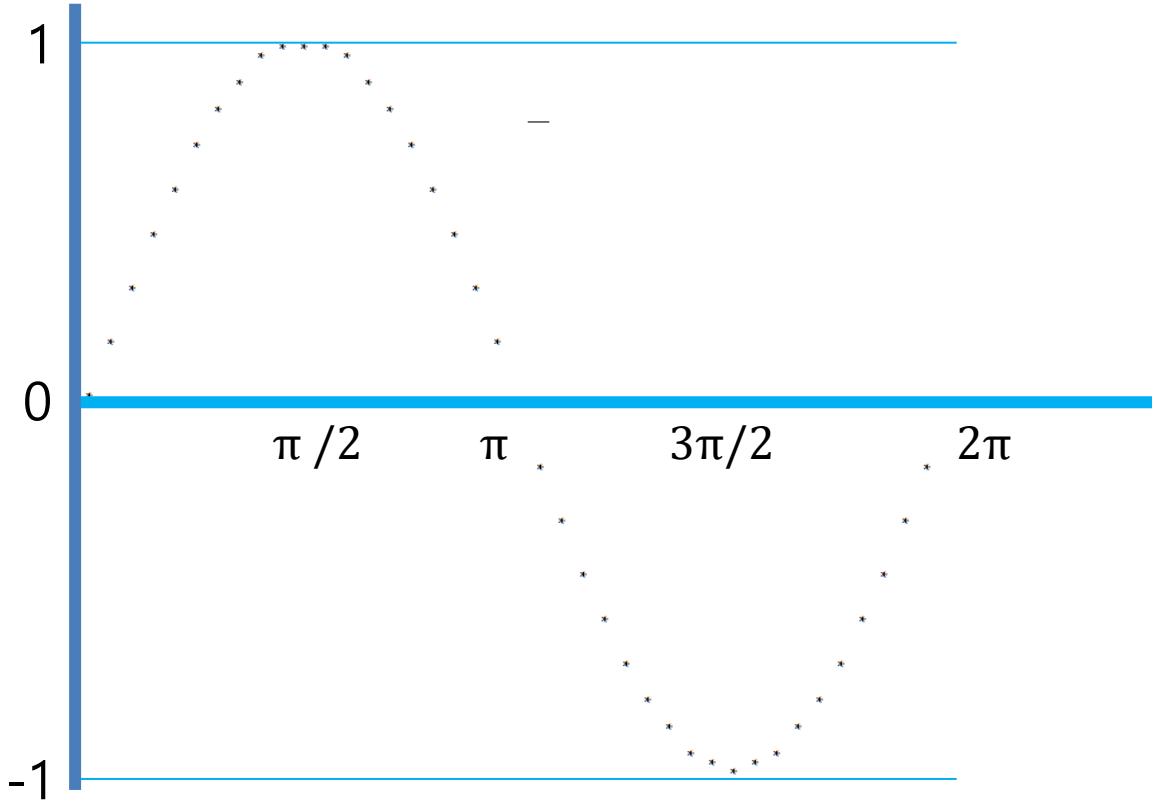
```
def take-numbers():
    Fill in this box.
    return x, y, z    # three number are assigned to x, y, z.

def check_inequality(x1, x2, x3):
    if x1 + x2 > x3 and x2 + x3 > x1 and x3 + x1 > x2:
        return True
    else:
        return False
```

DRAWING GRAPHS

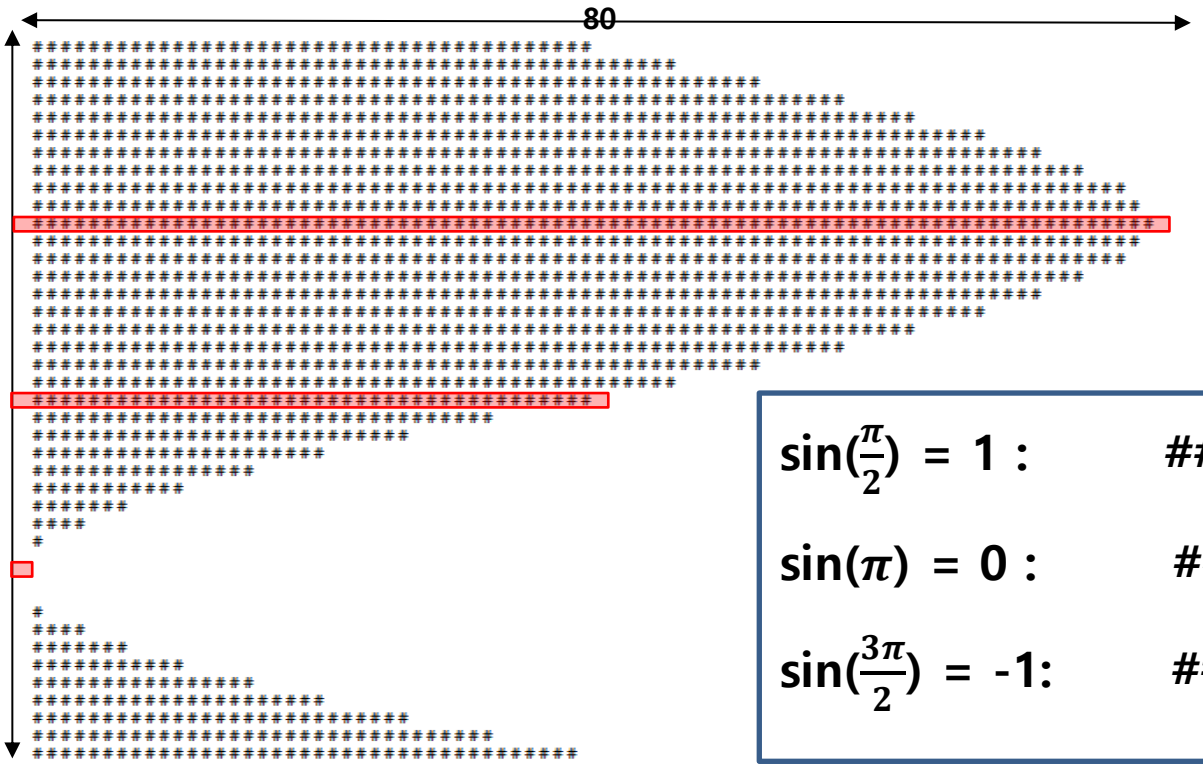
PROBLEM 3: DRAWING SIN CURVES

A triangular function **sin(angle)** gives a value between -1 and 1, inclusively depending on angle in radians as shown in the graph in the next slide. This graph shows how sin(angle) changes as angle varies from 0 to 2π. You are asked to write a program that plots the curve in two different forms: a bar graph and a point graph with axes.

Sine curve



1. Bar graph with #'s



$\sin(\frac{\pi}{2}) = 1$: ## · · · ##

80

$\sin(\pi) = 0$: ## · · · ##

40

$\sin(\frac{3\pi}{2}) = -1$: ## · · · ##

0

How to compute the number of #'s

$\sin(\frac{\pi}{2}) = 1$: ## · · · ##

80

$\sin(\pi) = 0$: ## · · · ##

40

$\sin(\frac{3\pi}{2}) = -1$: ## · · · ##

0

the number of #'s = sin(angle) * 40 + 40

Pseudo code

1. Change angle from 0 to 2 * π in k steps.
2. For each angle, compute the number of #'s and print the computed number of #'s

Employ for_loop to change the angle.

How to determine the number of steps k?

By trial and error! Try 40 steps.

Main program

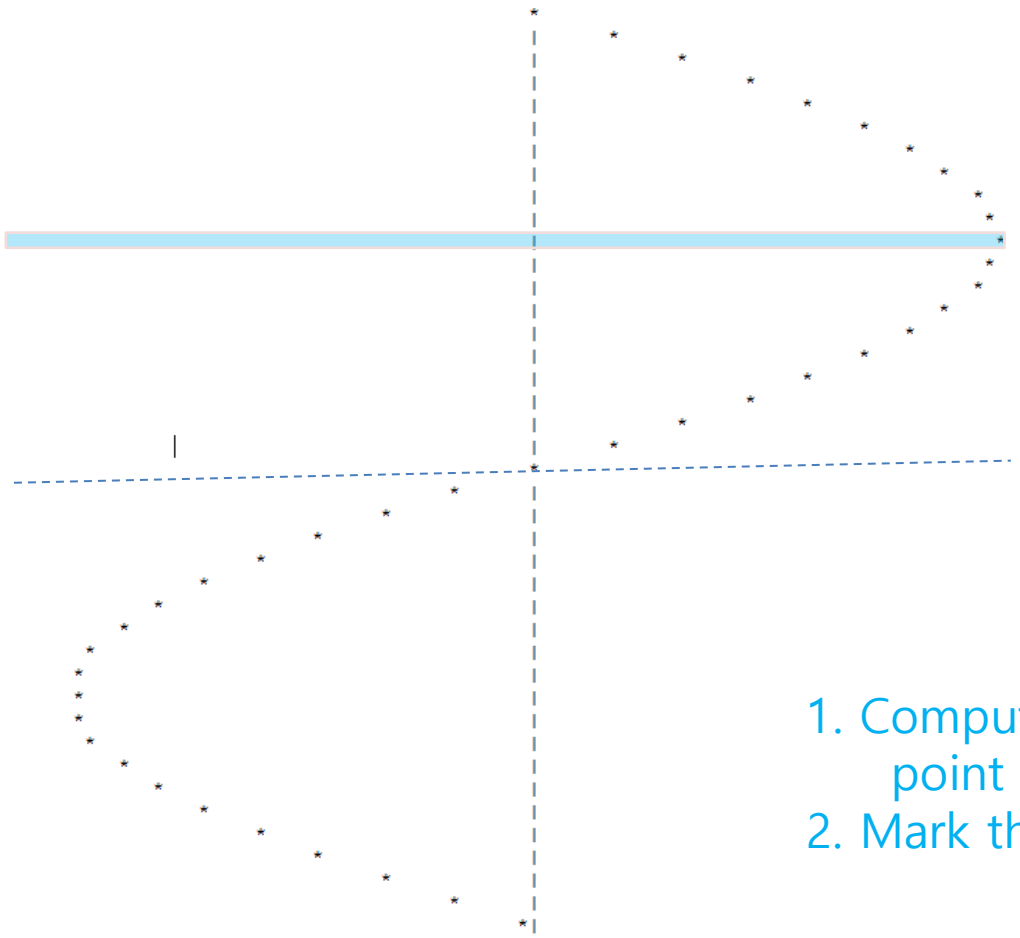
```
import math
for i in range(41):
    angle = (2 * math.pi / 40) * float(i)
    compute_and_plot(angle)
```

Compute and plot

compute_and_plot(x):

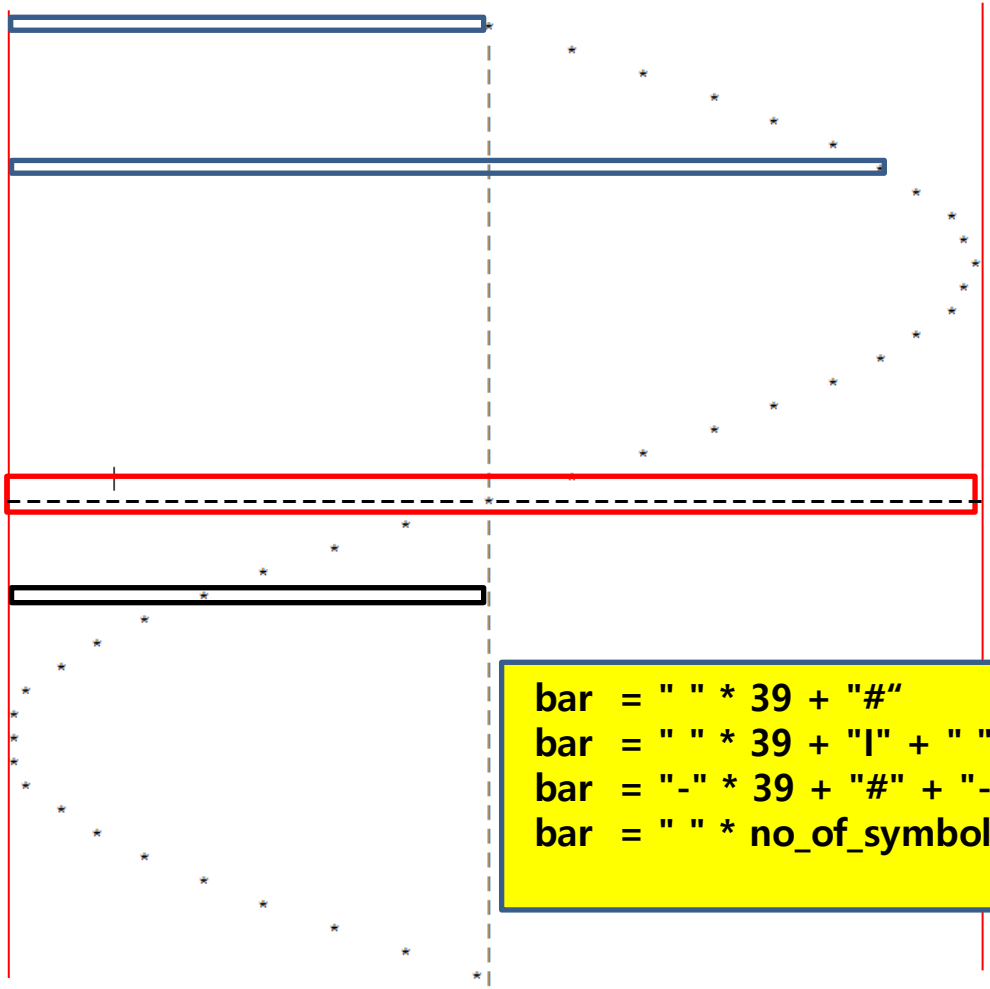
Compute the number of #'s for x.

Print the computed number of #'s.



1. Compute a bar and mark the point on the curve.
2. Mark the points on the axes.

How to compute a bar and mark the point on the curve



```
bar  = " " * 39 + "#“
bar  = " " * 39 + "I" + " " * (no_of_symbols - 40) + "#“
bar  = "-" * 39 + "#" + "-" * 40
bar  = " " * no_of_symbols  + "#" + " " * (38 - no_of_symbols ) + "I"
```