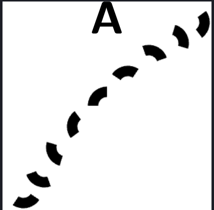
Announcements

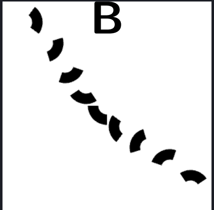
- Homework
  - HW2 due tonight!
  - Remember to change the “Assignment status:” to DONE!
  - I’m aiming to get HW3 posted tomorrow or early Sunday
- Polling: `rembold-class.ddns.net`

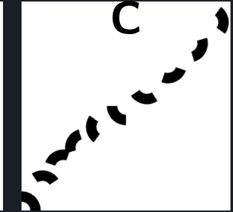
Review Question

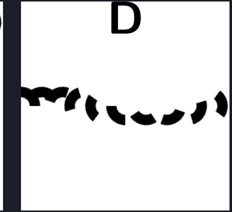
Suppose I included all the necessary code to import arcade, create a window of `WIDTH` and `HEIGHT`, run it, etc. Inside the start and finish render portions, I include the below code. What image would be drawn to the window?

```
i = 0
while i < WIDTH:
    A = i / WIDTH * 360
    arcade.draw_arc_outline(i, i, 20, 20, arcade.color.BLACK,
                            A, A+90, 15)
    i = i + 30
```


A


B


C


D

Moving On

- For the next 2 weeks we'll focus on common algorithms or approaches to solving problems
- Most code (with one exception) will mostly just use the pieces we've already learned
- Algorithms will mostly focus on trying to answer more numeric types of problems

Guess and Check the Python Way

- Suppose we were tasked with solving the (very simple) problem

$$(x + 12)^2 = 36$$

where we wanted to find any integer solutions of $\mathtt{x}$.

Guess and Check the Python Way

- Suppose we were tasked with solving the (very simple) problem

  $$(x+12)^2 = 36$$

  where we wanted to find any integer solutions of `x`.
- Instead of doing math, we could "guess" values of `x` and then check to see if we got the correct answer

Guess and Check the Python Way

- Suppose we were tasked with solving the (very simple) problem

  $$(x + 12)^2 = 36$$

  where we wanted to find any integer solutions of `x`.
- Instead of doing math, we could "guess" values of `x` and then check to see if we got the correct answer
- This would be a terrible way for you to solve the problem, but computers are king of solving tedious problems

Guess and Check the Python Way

- Suppose we were tasked with solving the (very simple) problem

$$(x + 12)^2 = 36$$

  where we wanted to find any integer solutions of `x`.
- Instead of doing math, we could "guess" values of `x` and then check to see if we got the correct answer
- This would be a terrible way for you to solve the problem, but computers are king of solving tedious problems
- Looking for solutions in this way is called exhaustive enumeration

Why guess?

- No doubt, exhaustive enumeration is usually not the most efficient way to find an answer, so why do it?
  - Algorithm is easy to read and understand
  - Tends to be simple to implement
  - Can easily give whole sets of solutions that meet conditions
  - Modern computers are **fast**
    - No sense spending hours coming up with a super efficient algorithm if it only saves you 5 milliseconds on runtime

Exhaustive Tips

- Important pieces to consider before using exhaustive enumeration:
  - You need to be able to (in theory) list out every single possible value that might be solution
    - Mathematically *countable*
    - Integers are a good example, as you could list them all out (even a huge number of them)
    - Real numbers or floats would *not* work, as they are uncountable
  - You need to be able to come up with a loop conditional which assures that your loop will end, even if a solution is not found.

Understanding Check

Which of the following questions would *not* be a suitable candidate for using some sort of exhaustive enumeration type algorithm?

A) Determining if a 3 digit number is prime

B) Checking if a particular substring exists within a larger string

C) Given 3 nickels, 2 dimes, and 8 pennies, finding all possible combinations totalling 25 cents

D) Finding all values of a and b where $a^2 + b^2 = 250$

Example 1: 3 digit prime?

- Countable solution set?
  - For any number (n), we would just need to check all the values smaller than that number to see if they divide the larger number equally
  - Set of values to check from $2 \rightarrow n - 1$, so countable
- Terminating condition?
  - Start at `x=2`
  - Stop when `x == n` (yes, there are more optimal ways)

Example 2: Substring exists?

- Countable solution set?
  - Just need to check each piece of the bigger string.
  - It is made up of a countable number of parts, so we just need to check a countable number
- Terminating Condition?
  - Assume `i=0` initially and increments by one
  - Taking the big string to be `b_str` and the substring to be `l_str`, keep going until

```
i + len(l_str) > len(b_str)
```

Example 3: Money Money

- Countable solution set?
  - We have a limited amount of coins to check
  - Just need to check all possible combinations
- Terminating Condition(s)?
  - Probably want one for each loop here
  - Assume `num_nic`, `num_dime`, `num_pen` all start at zero and increment by one
  - Stop when

```
 num_nic > max_nic
num_dime > max_dime
 num_pen > max_pen
```