Announcements

- Homework
  - I'm still working on getting all of HW2 scored, but hopefully by the end of today!
  - Homework 3 has been posted! Should be good to do Problems 1 and 2 after today.
- Polling: `rembold-class.ddns.net`

Review Question

Which of the following questions would be a suitable candidate for using some sort of exhaustive enumeration type algorithm?

A) Finding all multiples of 3 that are also divisible by 5
B) Counting the number of vowels in a sentence
C) Determining the precise amount of time to fire a rockets thrusters to reach a certain speed
D) Calculating the cubic root of 50

Solution: Counting the number of vowels in a sentence

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
```

Iterating over sequences

- It comes up often that we iterate over some sequence or range

```
i = 0
while i < 10:
    print(i)
    i = i + 1
```

- Python provides us with a simplified type of loop to do these sorts of iterations
- Called a for loop
  - Can think of it as saying: "For each thing in this stuff, run this code."

Anatomy of a For Loop

- For loops have a simple syntax:

```
for <variable> in <sequence>:
    <loop code>
```

Anatomy of a For Loop

- For loops have a simple syntax:

```
for <variable> in <sequence>:
    <loop code>
```

- <loop code> is no different from what we have already been doing inside loops

Anatomy of a For Loop

- For loops have a simple syntax:

```
for <variable> in <sequence>:
    <loop code>
```

- `<loop code>` is no different from what we have already been doing inside loops
- `<variable>` is the variable name that will take on every value in the sequence over the course of the loop

Anatomy of a For Loop

- For loops have a simple syntax:

```
for <variable> in <sequence>:
    <loop code>
```

- `<loop code>` is no different from what we have already been doing inside loops
- `<variable>` is the variable name that will take on every value in the sequence over the course of the loop
- `<sequence>` is a non-scalar object who individual pieces will be looped over

Simple String Example

- We already know strings are non-scalar objects, so they can be looped over with for loops!

```
dessert = "Pumpkin Pie"
for letter in dessert:
    print(letter)
```

- This is functionally identical to:

```
dessert = "Pumpkin Pie"
i = 0
while i < len(dessert):
    print(dessert[i])
    i = i + 1
```

Far Ranging Sequences

- How can this be useful with numbers?
- Python's `range()` function
  - Can pass in the same options as slicing
    - Start (defaults to 0)
    - Stop
    - Step (defaults to 1)
  - Like slicing, the `Stop` element will not be included
  - Unlike slicing though, use a comma (`,`) to separate the options!!
  - Generates values "as needed", so very memory efficient

For Ranging Examples

- Providing just a stop

```
for n in range(5):
    print(n)
```

- Providing start and stop

```
for n in range(1,11):
    print(n)
```

- Providing start, stop and step

```
for n in range(10,0,-1):
    print(n)
```

Understanding Check

Which of the below blocks of code would print something different than the others?

A)
```
for n in range(10):
    if n % 2 == 0:
        print(n)
```

B)
```
for i in range(0,10,2):
    if i:
        print(i)
```

C)
```
for j in range(0,20,4):
    print(j // 2)
```

D)
```
for k in range(0,10):
    if not k % 2:
        print(k)
```

Solution: B