Eric Roberts
CSCI 121

Handout #21
September 28, 2018

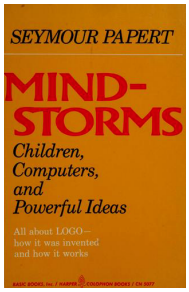
# String Applications

## String Applications

Eric Roberts
CSCI 121
September 28, 2018

## The Project LOGO Turtle

- In the late 1960s, Professor Seymour Papert at MIT developed the Project LOGO turtle and began using it to teach schoolchildren how to program.
- Papert described his experiences and his theories about education in a book entitled *Mindstorms,* which remains one of the most important books about computer science pedagogy.
- The Project LOGO turtle gave rise to the `TurtleGraphics` programming model, which in many of its versions is based entirely on strings.

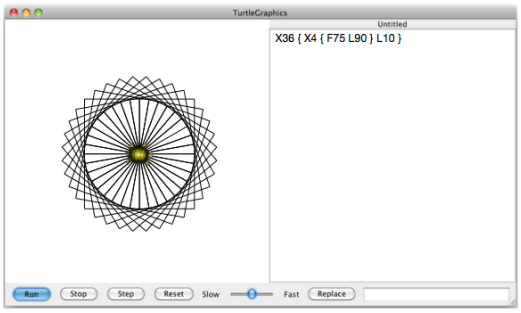



## `TurtleGraphics` Programs

- In string-based models of `TurtleGraphics`, programs are represented as strings composed of the following commands:

| | |
|---|---|
| **F***n* | Move the turtle forward *n* units |
| **L***n* | Turn left *n* degrees |
| **R***n* | Turn right *n* degrees |
| **U** | Raise the pen to stop line drawing |
| **D** | Lower the pen to resume line drawing |
| **X***n* **{***commands***}** | Repeat the specified commands *n* times |

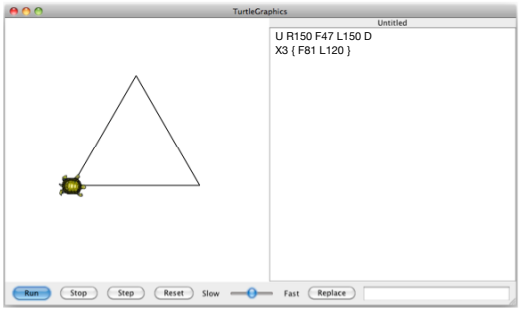
## TurtleGraphics in Action



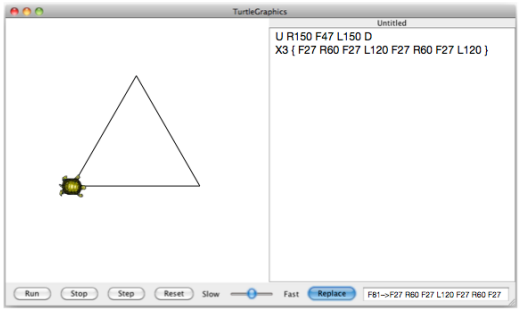

## Lindenmayer Systems

- Programs written using the string-based `TurtleGraphics` model evolve through repeated application of substitution patterns.
- Systems that evolve in this way are called ***L-systems*** or ***Lindenmayer systems***, after the Hungarian biologist Aristid Lindenmayer, who designed them to simulate plant growth.

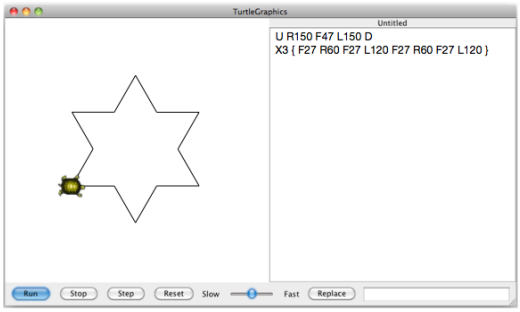
## Evolution through Substitution



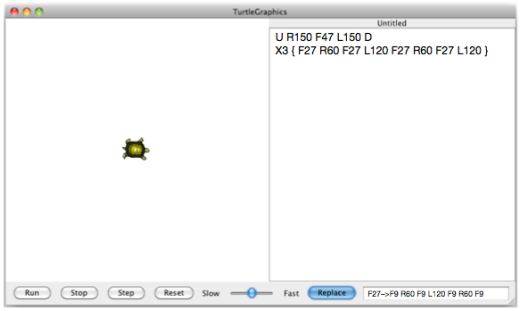
## Evolution through Substitution



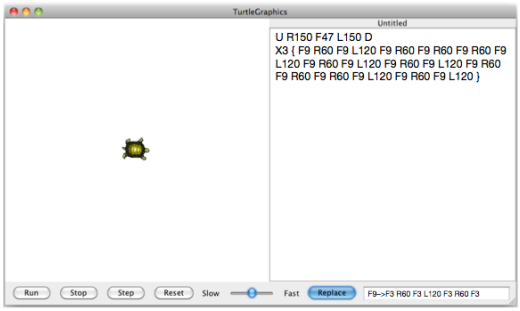

## Evolution through Substitution



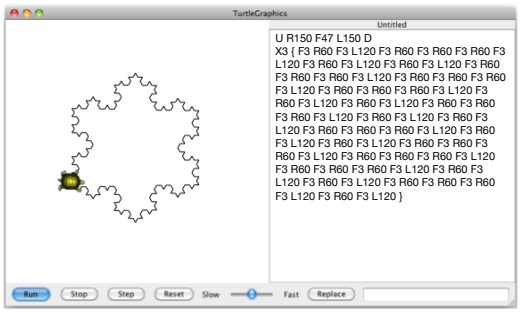

## Evolution through Substitution




## Evolution through Substitution




## Evolution through Substitution




## Generating Acronyms

- An ***acronym*** is a word formed by taking the first letter of each word in a sequence, as in

  `"port out starboard home"` → `"posh"`
  `"not in my back yard"` → `"nimby"`
  `"self-contained underwater breathing apparatus"` → `"scuba"`

- The text describes and implements two versions of a function `acronym(s)` that generates an acronym for `s`:
  - The first version searches for spaces in the string and includes the following character in the acronym. This version, however, fails for acronyms like *scuba,* in which some of the words are separated by hyphens rather than spaces.
  - The second version looks at every character and keeps track of whether the algorithm is scanning a word formed composed of sequential letters. This version correctly handles *scuba* as well as strings that have leading, trailing, or multiple spaces.

## First Version of the `acronym` Function

```
def acronym(s):
    result = s[0]
    sp = s.find(" ")
    while sp != -1:
        result += s[sp + 1]
        sp = s.find(" ", sp + 1)
    return result
```

| s | result | sp |
|---|---|---|
| | | |

```
TestAcronym
Enter a string: Laughing Out Loud
LOL
Enter a string: Read-Only Memory
RM
```

## Improved `acronym` Function

```
def acronym(s):
    result = ""
    inWord = False
    for ch in s:
        if ch.isalpha():
            if not inWord:
                result += ch
                inWord = True
        else:
            inWord = False
    return result
```

| s | result | ch | inWord |
|---|---|---|---|
| | | | |

```
TestAcronym
Enter a string: Read-Only Memory
ROM
```

## Translating Pig Latin to English

Section 6.5 works through the design and implementation of a program to convert a sentence from English to Pig Latin. In this dialect, the Pig Latin version of a word is formed by applying the following rules:

1. If the word begins with a consonant, the `wordToPigLatin` function moves the initial consonant string to the end of the word and then adds the suffix *ay*, as follows:

   *scram* ⟶ *amscr*

2. If the word begins with a vowel, `wordToPigLatin` generates the Pig Latin version simply by adding the suffix *way*, like this:

   *apple* ⟶ *appleway*

3. If the word contains no vowels at all, `wordToPigLatin` returns the original word unchanged.

## Pseudocode for the Pig Latin Program

```
def toPigLatin(str):
    Initialize a variable called result to hold the growing string.
    for each character position in str:
        if the current character is a letter:
            if we're not yet scanning a word:
                Remember the start of this word.
        else:
            if we're scanning a word:
                Call wordToPigLatin to translate the word.
                Append the translated word to the result variable.
            Append the separator character to the result variable.
    if we're still scanning a word:
        Call wordToPigLatin and append the translated word to result.

def wordToPigLatin(word):
    Find the first vowel in the word.
    If there are no vowels, return the original word unchanged.
    If the vowel appears in the first position, return the word concatenated with "way".
    Divide the string into two parts (head and tail) before the vowel.
    Return the result of concatenating the tail, the head, and the string "ay".
```

## Simulating the PigLatin Program

```
def toPigLatin(line):
    result = ""
    start = -1
    for i in range(len(line)):
        ch = line[i]
        if ch.isalpha():
            if start == -1:
                start = i
        else:
            if start >= 0:
                result += wordToPigLatin(line[start:i])
            start = -1
            result += ch
    if start >= 0:
        result += wordToPigLatin(line[start:])
    return result
```

| line | result | start | i | ch |
|---|---|---|---|---|
| | | | | |

```
IDLE
>>> toPigLatin("this is pig latin")
```

## Simulating the PigLatin Program

```
def toPigLatin(line):
def wordToPigLatin(word):
    vp = findFirstVowel(word)
    if vp == -1:
        return word
    elif vp == 0:
        return word + "way"
    else:
        head = word[0:vp]
        tail = word[vp:]
        return tail + head + "ay"
```

| word | vp | head | tail |
|---|---|---|---|
| | | | |

```
IDLE
>>> toPigLatin("this is pig latin")
isthay isway igpay atinlay
```