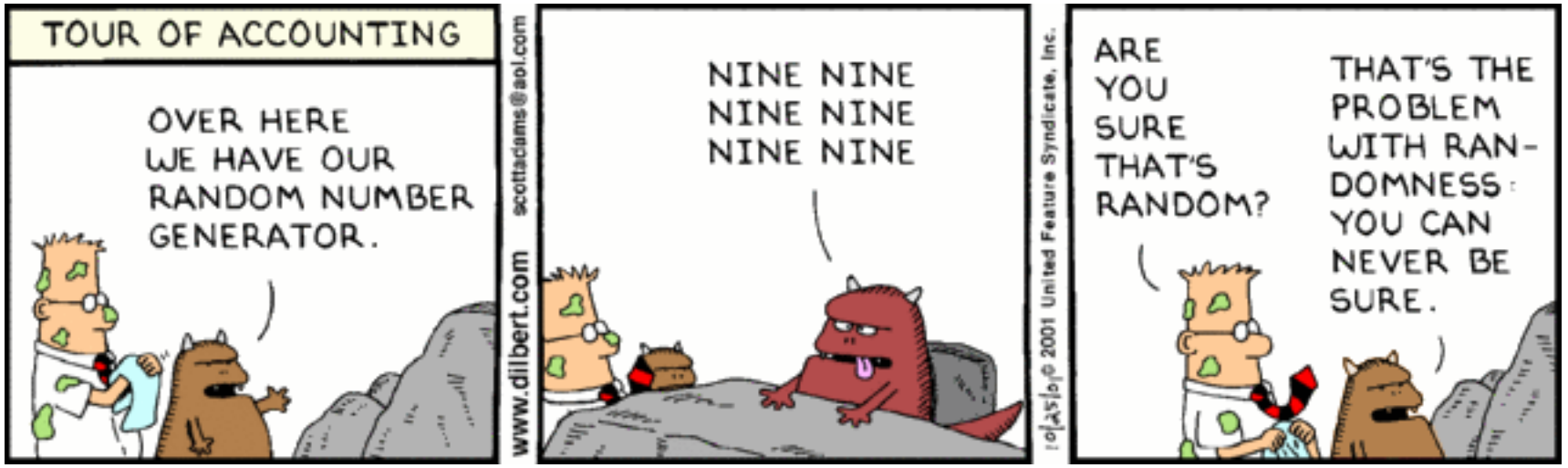
CSc 110, Autumn 2017

Lecture 15: Strings and Fencepost Loops

Adapted from slides by Marty Stepp and Stuart Reges




```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

http://xkcd.com/221/

Looping through a string

- The `for` loop through a string using range:

```
major = "CSc"
for letter in range(len(major)):
    print(major[letter])
```

- You can also use a `for` loop to print or examine each character without range.

```
major = "CSc"
for letter in major:
    print(letter)
```

Output:

```
C
S
c
```

`String` tests

Function	Description
`startswith(`**str**`)`	whether one contains other's characters at start
`endswith(`**str**`)`	whether one contains other's characters at end

```
name = "Voldermort"
if name.startswith("Vol"):
    print("He who must not be named")
```

- The `in` keyword can be used to test if a string contains another string.

  example: `"er" in name` `# true`

`String` question

- A *Caesar cipher* is a simple encryption where a message is encoded by shifting each letter by a given amount.
  - e.g. with a shift of 3, A → D, H → K, X → A, and Z → C

- Write a program that reads a message from the user and performs a Caesar cipher on its letters:

```
Your secret message: Brad thinks Angelina is cute
Your secret key: 3
The encoded message: eudg wklqnv dqjholqd lv fxwh
```

`Strings` and `ints`

- All `char` values are assigned numbers internally by the computer, called *ASCII* values.
  - Examples:
    `'A'` is 65, `'B'` is 66, `' '` is 32
    `'a'` is 97, `'b'` is 98, `'*'` is 42
  - One character long `Strings` and `ints` can be converted to each other
    `ord('a')` is 97, `chr(103)` is 'g'
  - This is useful because you can do the following:
    `chr(ord('a' + 2))` is `'c'`

A deceptive problem…

- Write a method `print_letters` that prints each letter from a word separated by commas.

  For example, the call:

  ```
  print_letters("Atmosphere")
  ```

  should print:

  ```
  A, t, m, o, s, p, h, e, r, e
  ```

Flawed solutions

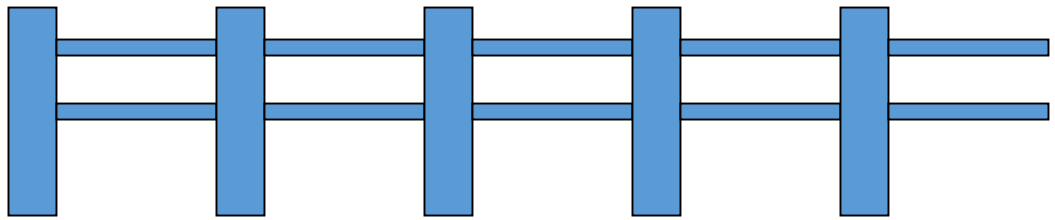
- ```
  def print_letters(word):
      for i in range(0, len(word)):
          print(word[i] + ", ", end='')
      print()    # end line
  ```
  - Output: A, t, m, o, s, p, h, e, r, e,

- ```
  def print_letters(word):
      for i in range(0, len(word)):
          print(", " + word[i], end='')
      print()    # end line
  ```
  - Output: , A, t, m, o, s, p, h, e, r, e

Fence post analogy

- We print *n* letters but need only *n* - 1 commas.
- Similar to building a fence with wires separated by posts:
  - If we use a flawed algorithm that repeatedly places a post + wire, the last post will have an extra dangling wire.

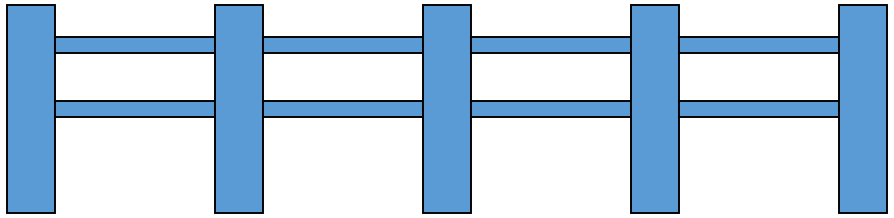
for length of fence :
    place a post.
    place some wire.



Fencepost loop

- Add a statement outside the loop to place the initial "post."
  - Also called a *fencepost loop* or a "loop-and-a-half" solution.

place a post.
for length of fence ***– 1****:*
    place some wire.
    place a post.



Fencepost function solution

- ```
  def print_letters(word):
      print(word[0])
      for i in range(1, len(word)):
          print(", " + word[i], end='')
      print()   # end line
  ```

- Alternate solution: Either first or last "post" can be taken out:

```
def print_letters(word):
    for i in range(0, len(word) - 1):
        print(word[i] + ", ", end='')
    last = len(word) - 1
    print(word[last]) # end line
```
```

Fencepost question

- Write a function `print_primes` that prints all *prime* numbers up to a max.
  - Example: `print_primes(50)` prints
    `2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47`
  - If the maximum is less than 2, print no output.

- To help you, write a function `count_factors` which returns the number of factors of a given integer.
  - `count_factors(20)` returns `6` due to factors 1, 2, 4, 5, 10, 20.

Fencepost answer

```
# Prints all prime numbers up to the given max.
def print_primes(max):
    if (max >= 2):
        print("2", end='')
        for i in range(3, max + 1):
            if (count_factors(i) == 2):
                print(", " + str(i))
        print()

# Returns how many factors the given number has.
def count_factors(number):
    count = 0
    for i in range(1, number + 1):
        if (number % i == 0):
            count += 1   # i is a factor of number
    return count
```