Objectives

- Escape sequences
- Computer's representations of data types

Mar 2, 2022 Sprenkle - CSCI111 1

1

Facebook Bug

Memory shown March 2, 2022.
Picture from March 2, 2020.

Your memories on Facebook

Sara, we care about you and the memories you share here. We thought you'd like to look back on this post from 1 year ago.

Mar 2, 2022 Sprenkle - CSCI111 2

2

Lab 6 Reflection

- Reflection: How far have I come in Computer Science?
- Indefinite loops require a different way of thinking
- Hardest problem was second rather than last
- Even more tools that you can combine—with new tools or old tools!
  - A lot of String operations
    - Previously: a lot of arithmetic operations, but you're familiar with those
- Break down problems
  - Solve what you can; break down what you can't
  - Not necessarily linear development
    - May do something and then undo it for the next step

3

Review

- How do you call a method on a string?
  - What is your favorite string method?
- True or False: You can change a string after it's been created

4

Review: Strings are Immutable

You cannot change the value of strings

- For example, you **cannot** change a character in a string
  - ~~str[0] = 'S'~~

Escape Sequences

- Escape character: `\`
- Escape sequences:
  - newline character (carriage return) → `\n`
  - tab → `\t`
  - quote → `\"` or `\'`
  - backslash → `\\`

Interactive demonstration

- Example:
  - `print("To print a \\, you must use \"\\\\\"")`
    - What does this display?

 demo_str.py

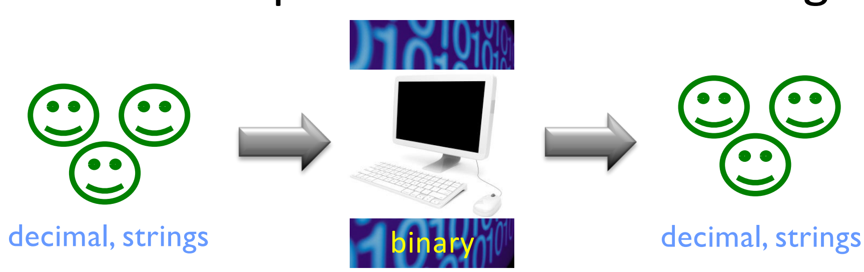
Practice

- Display To print a tab, you must use '\t'.
- Display I said, "How are you?"

`escape_sequence.py`

Representations of Data

- Computer needs to represent different types of data
  - Eventually, all boils down to 1s and 0s
- Computer needs to translate between what humans know to what computer knows and back again



Seems like a divergence on strings but just wait…

Decimal Representations

- Decimal is base 10
- Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Each *position* in a decimal number represents a power of 10

Decimal Representations

- Decimal is base 10
- Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Each *position* in a decimal number represents a power of 10
- Example: 54,087

5	4	0	8	7
10^4	10^3	10^2	10^1	10^0

- $= 5*10^4 + 4*10^3 + 0*10^2 + 8*10^1 + 7*10^0$
- = 5*10,000 + 4*1000 + 0*100 + 8*10 + 7*1

Number Representations

Characteristic	Decimal	Binary
Base	10	2
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0, 1
Position represents	Power of 10	Power of 2

- Binary: two values (0, 1)
  - Like a light switch (either **off** or **on**) or booleans (either True or False)
- 0 and 1 are *binary digits* or **bits**
  - 64-bit machine: represents numbers (and other data) with 64 bits

11

Binary Representation

- Binary number: 1101

1	1	0	1
2^3	2^2	2^1	2^0

- $= 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0$
- $= 1*8 + 1*4 + 0*2 + 1*1$
  - Decimal value: 13

Practice: what is the decimal value of the binary number **10110**?

12

Binary Representation

- Binary number: 10110

1	0	1	1	0
2^4	2^3	2^2	2^1	2^0

- $= 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0$
- $= 1*16 + 0*8 + 1*4 + 1*2 + 0*1$
  - 22

Generalize this process into an algorithm.
Implement as function: `binaryToDecimal(binaryNum)`
Define good test cases for this function.

Converting Binary to Decimal

- Generalize this process into an algorithm.
- Define good test cases for this algorithm/function
  - "Run" your algorithm on these test cases
- Implement as function:
  `binaryToDecimal(binaryNum)`

(Not necessarily sequential steps)

Algorithm 1: Converting Binary → Decimal

Left to right traversal of binary number

Accumulator design pattern

Given the binary number as a string

1. Initialize the result to zero
2. The starting exponent will be the length of the string-1
3. For each bit in the binary number
   - Multiply the bit by the appropriate power of 2
   - Add this to the result
   - Reduce the exponent by 1
4. Return the result

15

Algorithm 2: Converting Binary → Decimal

Right to left traversal of binary number

Accumulator design pattern

Given the binary number as a string

1. Initialize the result to zero
2. Initialize the exponent to zero
3. Iterate over the positions of the binary number from right to left
   - Determine the bit at that position in the binary number
   - Multiply the bit by the appropriate power of 2
   - Add this to the result
   - Increase the exponent by 1
4. Return the result

16

Practice

- Implement both algorithms
  - Test!

- After implementing, you can compare with my solutions
  - `binaryToDecimalIterateOverCharacters.py`
  - `binaryToDecimalIterateOverExponents.py`

Looking Ahead

- Lab 6 due Friday
- NLP Broader Issue – Thursday night