Objectives

- Strings
- Computer's representations of data types

Handshakes

- N=12 alumni

Sum these

11
10
9
8
7
6
5
4
3
2
1
0

```
11  10   9   8   7   6
 0   1   2   3   4   5
----------------------
11  11  11  11  11  11
```

Sums to 11 (N-1)
6 times (N/2)
= N*(N-1)/2

Handshakes

- N=13 alumni

Sum these

12
11
10
9
8
7
6
5
4
3
2
1
0

```
12  11  10   9   8   7
 1   2   3   4   5   6
----------------------
13  13  13  13  13  13
```

Sums to 13 (N)
6 times ((N-1)/2)
= N*(N-1)/2

Practice for Next Wed's Midterm

- Write a program that reads in two numbers. Then use only if statements (no elses) to print "Player 1 wins" if the first number is bigger, "Player 2 wins" if the second number is bigger, and "You tied!" if the numbers are equal.

Practice for Next Wed's Midterm

- Draw the control flow diagram for your Craps solution
  - To analyze efficiency: are there any execution paths through the control flow diagram that aren't possible?
    - If so, revisit your solution to see if some other building blocks may be more appropriate (or see me to discuss!)

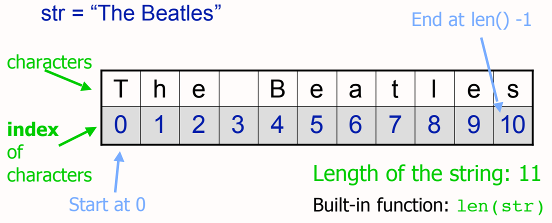
Text Processing

- Mostly focused on numbers so far
- We can manipulate strings to do useful work

- Focus: the **`str`** data type and what you can do with them
- Chapter 4 of book

Strings

- Actually a *sequence* of characters
  - Example:
    str = “The Beatles”



T	h	e		B	e	a	t	l	e	s
0	1	2	3	4	5	6	7	8	9	10

Length of the string: 11
Built-in function: `len(str)`
to find length of a string

Iterating Through a String

- Use a **`for`** loop to iterate through characters in a string

```
for char in str:
    print char
```

  - Read as “for each character in the string str”

Python shell

Substrings Operator

Literally, **not** optional

- Look at a particular character in the string
  - Syntax: string[<integer expression>]
  - [Positive values]: index of character
  - [Negative values]: count backwards from end
- Examples:
  - <sequence>[0] returns the first element/char
  - <sequence>[-1] returns the last element/char

We will deal with sequences beyond strings later.

Substrings Operator

- Look at a particular character in the string
  - Syntax: string[<integer expression>]
- Examples with str = “The Beatles”

Expression	Result
str[0]	
str[3]	
str[len(str)]	
str[len(str)-1]	
str[-1]	

Substrings Operator

- Look at a particular character in the string
  - Syntax: string[<integer expression>]
- Examples with str = “The Beatles”

Expression	Result
str[0]	“T”
str[3]	“ ”
str[len(str)]	IndexError
str[len(str)-1]	“s”
str[-1]	“s”

whilestr.py

Substrings Operator

- You can select a substring (zero or more characters) using the **[ ]** and **:**
- <sequence>[<start>:<end>]
  - returns the subsequence from **start** up to and not including **end**
- <sequence>[<start>:]
  - returns the subsequence from **start** to the end of the sequence
- <sequence>[:<end>]
  - returns the subsequence from the first element up to and not including **end**
- <sequence>[:]
  - returns a copy of the entire sequence

Substrings Operator

- You can select a substring (one or more characters) using the [ ] and :
- Examples: file = “program.py”

p	r	o	g	r	a	m	.	p	y
0	1	2	3	4	5	6	7	8	9

Expression	Result
file[0:]	
file[0:2]	
file[:3]	
file[8:]	
file[-2:]	

Substrings Operator

- You can select a substring (one or more characters) using the [ ] and :
- Examples: file = “program.py”

p	r	o	g	r	a	m	.	p	y
0	1	2	3	4	5	6	7	8	9

Expression	Result
file[0:]	“program.py”
file[0:2]	“pr”
file[:3]	“pro”
file[8:]	“py”
file[-2:]	“py”

Testing for Substrings

- Using the **in** operator
  - Used **in** before in for loops
- Syntax:

```
if substring in string:
```

- Evaluates to True or False
- Example:

```
if ".py" in filename:
        print filename, "is a Python script"
```

Strings are Immutable

- Note: You cannot change the value of strings
- For example, you cannot change a character in a string
  - ~~str[0] = ‘S’~~

Number Representations

- We briefly discussed that numbers are stored on the computer in binary
  - Binary representation
- Binary: two values (zero, one)
  - Like a light switch (either **off** or **on**)
- **Bit**: each number in a number in binary representation
  - Equivalent of a “digit” in decimal representation

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

Binary Representation

- Binary is base 2
- Digits: 0, 1
- Each position in a binary number represents a power of 2

Binary Representation

- Example: 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
  - 13
- Practice: 10110

Binary Representation

- Example: 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

Converting Binary to Decimal

- Accumulator design pattern
- Read in the binary number as a string
  - The starting exponent will be the length of the string-1
- Initialize the result to zero
- 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
- Print the result

Implement algorithm binaryToDecimal.py

Converting Decimal to Binary

- Read in the decimal as an integer
- Initialize the result to the empty string
- Repeat until the decimal is 0:
  - result = str(decimal % 2) + result
  - decimal = decimal / 2
- Print the result

Try out algorithm with 22
Implement algorithm decimalToBinary.py