Objectives

- Escape sequences
- Computer's representations of data types

1

Review

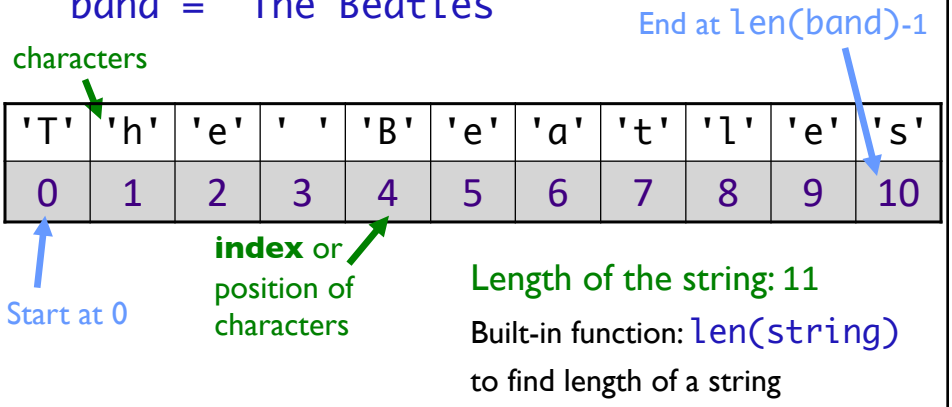
- How can we combine strings?
- How can we find out how long a string is?
  - How do we call it?
- How can you tell if one string is contained within another string?
- How can we find out the character at a certain position?
- How can we iterate through a string? (Two ways)
- How do you call a method on a string?
  - What is your favorite string method?
- True or False: You can change a string after it's been created

2

Review: Strings

- A ***sequence*** of one-character strings
  - Example:

```
band = "The Beatles"
```



Length of the string: 11

Built-in function: `len(string)` to find length of a string

3

Review: Iterating Through a String

- For each character in the string

string of length 1

```
for char in mystring:
    print(char)
```

Determines loop's behavior

- For each position in the string

An integer

```
for pos in range(len(mystring)):
    print(mystring[pos])
```

Index into the string

4

Review: Testing for Substrings

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

```
substring in string:
```

  - Evaluates to `True` or `False`
- Example:

```
if searchTerm in documentText:
      print(documentText, "contains", searchTerm)
```


5

Review: Strings are Immutable

You cannot change the value of strings

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

6

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?

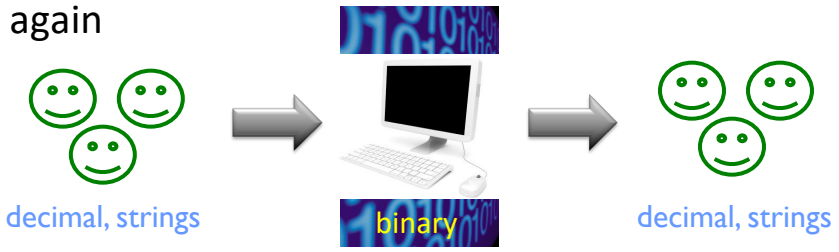
Practice

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

escape_sequence.py

Representations of Data

- Computer needs ways 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…

9

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

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

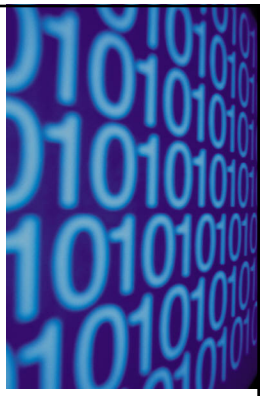
- $= 5*10^4 + 4*10^3 + 0*10^2 + 8*10^1 + 7*10^0$
- $= 5*10{,}000 + 4*1000 + 0*100 + 8*10 + 7*1$

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

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**?

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)`

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

Good test cases?

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

Good test cases?

16

Practice

- Implement both algorithms
  - Test!
- After implementing, you can compare with my solutions
  - `binaryToDecimalIterateOverCharacters.py`
  - `binaryToDecimalIterateOverExponents.py`

17

Algorithm: Converting Decimal → Binary

Given the decimal as an integer…

1. Initialize the result to the empty string
2. Repeat until the decimal is 0:
   - result = str(decimal % 2) + result
   - decimal = decimal // 2
3. Return the result

1. Try out algorithm with 22 as input
2. Implement algorithm in function `decimalToBinary`
3. Good test cases?

 `decimalToBinary.py`

18

Looking Ahead

- Pre Lab 7
  - Back to indefinite loops
  - More strings
    - Repeating section about string formatting (more on Monday)