# Objectives

- Escape Sequences
- Computer's representations of data types

# Lab 6 Reflection

- Reflection: How far have I come in Computer Science?
- Indefinite loops require a different way of thinking
- Likely, 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

# ESCAPE SEQUENCES

# Escape Sequences

- Escape sequences: Represent special characters within a string
  - Considered a single character
- Escape character: \
  - The character following the escape character tells you how to interpret the escape sequence

| Escape Sequence | Meaning |
|---|---|
| \n | Newline character (carriage return) |
| \t | Tab |
| \" or \' | Quote |
| \\ | Backslash |

# Using Escape Sequences

```
>>> mystr = "Hello!\nHi!"
>>> mystr
'Hello!\nHi!'
>>> print(mystr)
Hello!
Hi!
>>> len(mystr)
10
```

What do these evaluate to?

```
mystr[5]
mystr[6]
mystr[7]
```

# Using Escape Sequences

```
>>> mystr = "Hello!\nHi!"
>>> mystr
'Hello!\nHi!'
>>> print(mystr)
Hello!
Hi!
>>> len(mystr)
10
```

What do these evaluate to?

```
mystr[5] → "!"
mystr[6] → "\n"
mystr[7] → "H"
```

# Escape Sequences

- Escape sequences: Represent special characters within a string
  - Considered a single character
- Escape character: \
  - The character following the escape character tells you how to interpret the escape sequence
- Example:
  - `print("To print a \\, you must use \"\\\\\"")`
    - What does this display?

| Escape Sequence | Meaning |
|---|---|
| \n | Newline character (carriage return) |
| \t | Tab |
| \" or \' | Quote |
| \\ | Backslash |

Interactive demonstration

demo_str.py

# Review: Description of `print`

- `print(*objects, sep=' ', end='\n', file=sys.stdout)`

Semantics: default values for `sep` is `' '` and `end` is `'\n'`

- Print *object*(s) to the stream *file*, separated by *sep* and followed by *end*.
- Both *sep* and *end* must be strings; they can also be `None`, which means to use the default values. If no *object* is given, *print*() will just write *end*.

 `https://docs.python.org/3/library/functions.html#print`

# Practice

- What does this output?

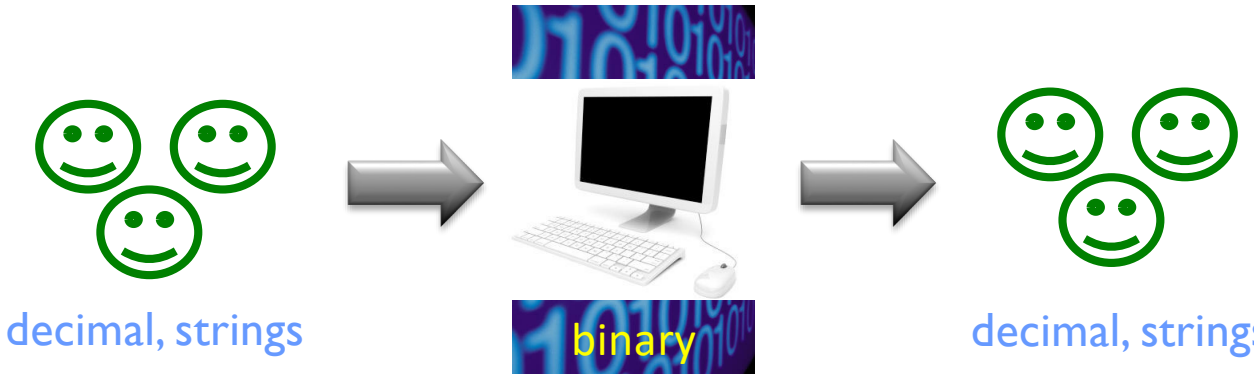
```
print("\n")
```

- Display To print a tab, you must use '\t’.

- Display I said, "How are you?"

# 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

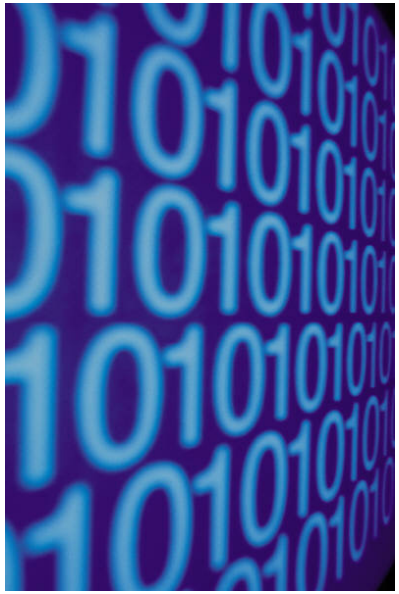
| Characteristic | Decimal | Binary |
|---|---|---|
| **Base** | 10 | 2 |
| **Digits** | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 | 0, 1 |
| **Position represents** | Power of 10 | Power of 2 |

- 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

# 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

# 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

# Process to Implement Binary to Decimal Conversion

1. Design function API: `binary_to_decimal(binary_num)`
   - Takes as parameter the binary number as a *string* (Why?)
   - Returns the decimal value of the given binary number
2. Define good test cases for this function
   - Input, expected results
3. Generalize this process into an algorithm
4. "Run" your algorithm on these test cases
5. Implement your algorithm

## 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

## 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

# Practice

- Implement both algorithms
  - Test!

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

# Converting Decimal → Binary

- What should the function API be?
- Define test cases

# 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 `decimal_to_binary`

# Looking Ahead

- Lab 6 due Friday
- Broader Issue: Social Media Company Responsibility