Preliminaries: Announcements

Homework #1

• Do not leave to the last minute. Start if you haven’t yet started!
• Refer to the lecture slides and e-book to answer the 10 questions
• Refer to the lecture slides and lab 1 for hints on the single programming task

• You CAN (you are encouraged, actually) to work with other students, but remember that you can only discuss the homework. You can jot on the white board, have a group discussion, etc., but the code file that you submit must be your own.

Monday, 18 January

• No lecture
Preliminaries : Announcements

Two workshops focused on Women in CS
21 January, 5-7pm
28 January, 5-7pm
Seats are limited

To register:
http://www.signupgenius.com/go/10c0e45a4ac2aabf49-lilblumes/
Preliminaries: Announcements

## TAs and Filip’s office hours, Tutoring

<table>
<thead>
<tr>
<th>Time</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>11am-noon</td>
<td>Filip, CF461</td>
<td>Filip, CF461</td>
<td>Filip, CF461</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2pm-4pm</td>
<td></td>
<td>Albert, CF405</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4pm-5pm</td>
<td>Robert, CF418</td>
<td></td>
<td>Robert, CF418</td>
<td>Robert, CF418</td>
<td>Robert, CF418</td>
</tr>
<tr>
<td>4pm-7pm</td>
<td>ACM Tutoring, CF162/164</td>
<td>ACM Tutoring, CF162/164</td>
<td>ACM Tutoring, CF162/164</td>
<td>ACM Tutoring, CF162/164</td>
<td>ACM Tutoring, CF162/164</td>
</tr>
</tbody>
</table>
From Last Time

Q: What is the value assigned to the variable muchNum after the code in the box is executed?

```
muchNum = 5 % (3 ** ( 6 // 4))
```

A. 0  
B. 1  
C. 2  
D. 3  
E. 4  
F. 5  
G. 7653  
H. -44
Q: What is the value assigned to the variable muchNum after the code in the box is executed?

A. 0  
B. 1  
C. 2  
D. 3  
E. 4  
F. 5  
G. 7653  
H. -44

Step 1: evaluate the inner-most arithmetic first

\[
\text{muchNum} = 5 \% (3 ** (6 // 4))
\]

\[
\text{muchNum} = 5 \% (3 ** (1))
\]

6 // 4 = 1
Q: What is the value assigned to the variable `muchNum` after the code in the box is executed?

A. 0
B. 1
C. 2
D. 3
E. 4
F. 5
G. 7653
H. -44

Step 2: evaluate the next inner-most arithmetic

```
muchNum = 5 % (3 ** (6 // 4))
muchNum = 5 % (3 ** (1))
muchNum = 5 % (3)
```

```
3 ** 1 = 3
```

A. 0
From Last Time

Q: What is the value assigned to the variable `muchNum` after the code in the box is executed?

A. 0  
B. 1  
C. 2  
D. 3  
E. 4  
F. 5  
G. 7653  
H. -44

Step 3: evaluate the “last”, or outer-most expression

3 ** 1 = 3

muchNum = 5 % (3 ** (1))

muchNum = 5 % (3)

muchNum = 2
Today

Keywords
Order of Operations
Assignment Operator
Binary, Decimal conversion
ASCII
Debugging
The Coding Process
We’ve already briefly discussed “good” variable names. Camel capitalization or camel case is a good approach to make compound variables more easily readable. But that’s a convention.

Q: Are there “Rules” for naming variables?

Q: Can you name your variable anything you want?
Keywords

We’ve already briefly discussed “good” variable names. Camel capitalization or camel case is a good approach to make compound variables more easily readable. But that’s a convention.

Q: Are there “Rules” for naming variables?

- A variable **must** begin with a letter or underscore. It cannot begin with a number
Keywords

We’ve already briefly discussed “good” variable names. Camel capitalization or camel case is a good approach to make compound variables more easily readable. But that’s a convention.

Q: Are there “Rules” for naming variables?

• A variable **must** begin with a letter or underscore. It cannot begin with a number
• A variable name cannot be the same as an already existing function or a construct that refers to Python’s syntax. These are called **keywords**, and you cannot name a variable as one of these. The keywords are:
  and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, nonlocal, not or, pass, raise, return, try, while, with, yield, True, False, None
We’ve already briefly discussed “good” variable names. Camel capitalization or camel case is a good approach to make compound variables more easily readable. But that’s a convention.

Q: Are there “Rules” for naming variables?

- A variable **must** begin with a letter or underscore. It cannot begin with a number.
- A variable name cannot be the same as an already existing function or a construct that refers to Python’s syntax. These are called **keywords**, and you cannot name a variable as one of these. The keywords are:
  - and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, nonlocal, not or, pass, raise, return, try, while, with, yield, True, False, None
- Another convention is that variable name begin with lower case letters (later you’ll learn that variable names that begin with a Capital letter refer to an instance of a specific object)
Order of Operations

We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

If the parentheses in today’s original expression were removed ...

```
muchNum = 5 % (3 ** ( 6 // 4))
```

```
muchNum = 5 % 3 ** 6 // 4
```

(on the board explanation)
We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

- Parentheses
- Exponentiation
- Multiplication and Division
- Addition and Subtraction
We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

```
6 * 6 ** 2 / 5 // 9 - 4
```

**Q: What does this expression evaluate to?**
We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parentheses</td>
<td><strong>( )</strong></td>
<td></td>
</tr>
<tr>
<td>Exponentiation</td>
<td><strong>^</strong></td>
<td>$6 \times 6^2 \div 5 \div 9 - 4$</td>
</tr>
<tr>
<td>Multiplication and Division</td>
<td><strong>\times, \div</strong></td>
<td></td>
</tr>
<tr>
<td>Addition and Subtraction</td>
<td><strong>+,-</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Q: Does this expression contain parentheses?**

6 * 6 ** 2 / 5 // 9 - 4
We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

**Precedence**

1. Parentheses
2. Exponentiation
3. Multiplication and Division
4. Addition and Subtraction

**Example:**

6 * 6 ** 2 / 5 // 9 - 4

**Q: Does this expression contain parentheses?** No

**Q: Does this expression contain exponentiation?**
Order of Operations

We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

- Parentheses
- Exponentiation
- Multiplication and Division
- Addition and Subtraction

6 * 6 ** 2 / 5 // 9 - 4

**Q: Does this expression contain exponentiation?** Yes
We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parentheses</td>
<td>Exponentiation</td>
</tr>
<tr>
<td>Multiplication and Division</td>
<td>Addition and Subtraction</td>
</tr>
</tbody>
</table>

**Example:**

\[6 \times 6 \times 2 / 5 \div 9 - 4\]

**Q: Does this expression contain exponentiation?** Yes
Order of Operations

We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operation</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parentheses</td>
<td>6 * 6 ** 2 / 5 // 9 – 4</td>
</tr>
<tr>
<td></td>
<td>Exponentiation</td>
<td>6 * 36 / 5 // 9 – 4</td>
</tr>
<tr>
<td></td>
<td>Multiplication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>and Division</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Addition</td>
<td></td>
</tr>
<tr>
<td></td>
<td>and Subtraction</td>
<td></td>
</tr>
</tbody>
</table>

**Q: Does this expression contain multiplication or division?**
We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parentheses</td>
<td></td>
</tr>
<tr>
<td>Exponentiation</td>
<td>6 * 6 ** 2</td>
</tr>
<tr>
<td>Multiplication and Division</td>
<td>/ 5  // 9 - 4</td>
</tr>
<tr>
<td>Addition and Subtraction</td>
<td>6 * 36 / 5 // 9 - 4</td>
</tr>
</tbody>
</table>

**Q: Which of these is evaluated first?**

**Q: Does this expression contain multiplication or division?** Yes
Order of Operations

We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parentheses</td>
<td></td>
</tr>
<tr>
<td>Exponentiation</td>
<td></td>
</tr>
<tr>
<td>Multiplication and Division</td>
<td>6 * 6 ** 2 / 5 // 9 - 4</td>
</tr>
<tr>
<td>Addition and Subtraction</td>
<td>6 * 36 / 5 // 9 - 4</td>
</tr>
</tbody>
</table>

**Q: Which of these is evaluated first?**

**Q: Does this expression contain multiplication or division?** Yes
We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

**precedence**

- Parentheses
- Exponentiation
- Multiplication and Division
- Addition and Subtraction

6 * 6 ** 2 / 5 \ 9 - 4

6 * 36 / 5 \ 9 - 4

**Q: Which of these is evaluated first?**

**Q: Does this expression contain multiplication or division?** Yes
Order of Operations

We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

**Precedence**

- Parentheses
- Exponentiation
- Multiplication and Division
- Addition and Subtraction

**Expression:**

\[
6 \times 6^{2} / 5 \; \text{//} \; 9 - 4
\]

- \[
6 \times 36 / 5 \; \text{//} \; 9 - 4
\]
- \[
216 / 5 \; \text{//} \; 9 - 4
\]

**Evaluated Left-to-right**
We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parentheses</td>
<td></td>
<td>$6 \times 6 \times 2 / 5 \div 9 - 4$</td>
</tr>
<tr>
<td>Exponentiation</td>
<td></td>
<td>$6 \times 36 / 5 \div 9 - 4$</td>
</tr>
<tr>
<td>Multiplication and Division</td>
<td></td>
<td>$216 / 5 \div 9 - 4$</td>
</tr>
<tr>
<td>Addition and Subtraction</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parentheses</td>
<td>$6 \times 6^{2} / 5 \div 9 - 4$</td>
</tr>
<tr>
<td>Exponentiation</td>
<td>$6 \times 36 / 5 \div 9 - 4$</td>
</tr>
<tr>
<td>Multiplication and Division</td>
<td>$216 / 5 \div 9 - 4$</td>
</tr>
<tr>
<td>Addition and Subtraction</td>
<td>$43.2 \div 9 - 4$</td>
</tr>
</tbody>
</table>
We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parentheses</td>
<td>$(6 \times 6^2) / 5 \div 9 - 4$</td>
<td>$4.0$</td>
</tr>
<tr>
<td>Exponentiation</td>
<td>$6 \times 36 / 5 \div 9 - 4$</td>
<td>$4.0$</td>
</tr>
<tr>
<td>Multiplication and Division</td>
<td>$216 \div 5 \div 9 - 4$</td>
<td>$4.0$</td>
</tr>
<tr>
<td>Addition and Subtraction</td>
<td>$43.2 \div 9 - 4$</td>
<td>$4.0$</td>
</tr>
</tbody>
</table>
We’ve mentioned that expressions are evaluated from the inner-most parentheses outward. **Q: Are there other rules?**

Yes. There is an order of operations if more than one operator is in an expression. This order of operations is similar to what you might (hopefully?) remember from elementary school.

<table>
<thead>
<tr>
<th>precedence</th>
<th>Parentheses</th>
<th>Exponentiation</th>
<th>Multiplication and Division</th>
<th>Addition and Subtraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 * 6 ** 2 / 5 // 9 - 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 * 36 / 5 // 9 - 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>216 / 5 // 9 - 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43.2 // 9 - 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.0 - 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
On the Fly Q

Task: Place parentheses and the mathematical operators of your choice among the digits 7, 2, 51 and 11 so that the expression evaluates to 7 (you cannot place anything between digits of a multi-digit number)
Task: Place parentheses and the mathematical operators of your choice among the digits 7, 2, 51 and 11 so that the expression evaluates to 7 (you cannot place anything between digits of a multi-digit number)

\[(7 ** 2) // (51 \% 11)\]
On the Fly Q

Task: Place parentheses and the mathematical operators of your choice among the digits 7, 2, 51 and 11 so that the expression evaluates to 7 (you cannot place anything between digits of a multi-digit number)

\[ (7 ** 2) \div (51 \mod 11) \]

\[ (49) \div (7) \]

precedence

Parentheses

Exponentiation

Multiplication and Division

Addition and Subtraction
Task: Place parentheses and the mathematical operators of your choice among the digits 7, 2, 51 and 11 so that the expression evaluates to 7 (you cannot place anything between digits of a multi-digit number)

(7 ** 2) // (51 % 11)

(49) // (7)

7
Assignment Operator

All operators have a specific sequence of events that they follow. The assignment operator, =, is no exception.
Assignment Operator

All operators have a specific sequence of events that they follow. The assignment operator, =, is no exception.

```
numCows = 32
numCats = 4
numCows = numCats + 4
print(numCows)
```

Q: What is the output of the code on the left?

A. 2  
B. 4  
C. 8  
D. 32  
E. 36
Assignment Operator

All operators have a specific sequence of events that they follow.
The assignment operator, =, is no exception

<table>
<thead>
<tr>
<th>numCows = 32</th>
</tr>
</thead>
<tbody>
<tr>
<td>numCats = 4</td>
</tr>
<tr>
<td>numCows = numCats + 4</td>
</tr>
<tr>
<td>print(numCows)</td>
</tr>
</tbody>
</table>

Remember that a variable is a name that allows you to easily refer to the value of an object of a certain type
Assignment Operator

All operators have a specific sequence of events that they follow. The assignment operator, =, is no exception.

| numCows = 32 |
| numCats = 4 |
| numCows = numCats + 4 |
| print(numCows) |

Remember that a variable is a name that allows you to easily refer to the value of an object of a certain type.
Assignment Operator

All operators have a specific sequence of events that they follow. The assignment operator, =, is no exception.

Q: How does the assignment operator “work”?

```
numCows = 32
numCats = 4
numCows = numCats + 4
print(numCows)
```
Assignment Operator

All operators have a specific sequence of events that they follow. The assignment operator, =, is no exception.

```
numCows = 32
numCats = 4
numCows = numCats + 4
print(numCows)
```

Q: How does the assignment operator “work”?

The assignment operator first performs all “calculations” to the right of =.

In this example, Python would “go to” the value that is referred to as `numCats`, and “fetch” it, and perform the calculation `4 + 4`.

Important: The value of `numCats` is NOT altered because the variable is to the right of the assignment operator =.
Assignment Operator

All operators have a specific sequence of events that they follow. The assignment operator, =, is no exception.

```python
numCows = 32
numCats = 4
numCows = numCats + 4
print(numCows)
```

Q: How does the assignment operator “work”?

If a variable already exists, when you “update” its value, you are making the existing variable name refer to a new value (a new object).

The “old” object is no longer being referred to by the variable.
Assignment Operator

All operators have a specific sequence of events that they follow. The assignment operator, `=`, is no exception.

```plaintext
numCows = 32
numCats = 4
numCows = numCats + 4
print(numCows)
```

Q: How does the assignment operator “work”?

If a variable already exists, when you “update” its value, you are making the existing variable name refer to a new value (a new object).

The “old” object is no longer being referred to by the variable.

The process of “removing” from memory old data objects that are no longer being referred to is called **garbage collection**.
Assignment Operator

All operators have a specific sequence of events that they follow. The assignment operator, =, is no exception.

```
numCows = 32
numCats = 4
numCows = numCats + 4
print(numCows)
```

Q: How does the assignment operator “work”?  

The print function then “goes to” the value that is referred to as `numCows`, and prints that value to the output.

A. 2  
B. 4  
C. 8  
D. 32  
E. 36

Correct Answer: C. 8
On-the fly question

Q: What is the output of the code in the box?

```python
aNumber = 4
aNumber = 8
aNumber = aNumber + 6
print(aNumber)
```

A. 4  
B. 8  
C. 10  
D. 12  
E. 14  
F. 16  
G. None of the above
On-the-fly question

Q: What is the output of the code in the box?

```
aNumber = 4
aNumber = 8
aNumber = aNumber + 6
print(aNumber)
```

A. 4  
B. 8  
C. 10  
D. 12  
E. 14  
F. 16  
G. None of the above
On-the-fly question

Q: What is the output of the code in the box?

```
aNumber = 4
aNumber = 8
aNumber = aNumber + 6
print(aNumber)
```

A. 4  
B. 8  
C. 10 
D. 12 
E. 14 
F. 16 
G. None of the above
Q: What is the output of the code in the box?

```
aNumber = 4
aNumber = 8
aNumber = aNumber + 6
print(aNumber)
```

Calculation: $8 + 6 = 14$

A. 4  
B. 8  
C. 10  
D. 12  
E. 14  
F. 16  
G. None of the above
On-the-fly question

Q: What is the output of the code in the box?

```python
aNumber = 4
aNumber = 8
aNumber = aNumber + 6
print(aNumber)
```

A. 4  
B. 8  
C. 10 
D. 12 
E. 14 
F. 16 
G. None of the above
On-the fly question

Q: What is the output of the code in the box?

```python
aNumber = 4
aNumber = 8
aNumber = aNumber + 6
print(aNumber)
```

A. 4
B. 8
C. 10
D. 12
E. 14
F. 16
G. None of the above

A Number 14

4

8

14


We’ve starting learning about variables ... but two question not yet answered ... and something you should know HOW it works ...

Q: 1s and 0s?

Q: How is data stored in a computer’s memory?

(You’ll learn the answers to these questions in much more detail when you take a computer architecture course, but to satisfy your curiosity now ... )
Binary, Decimal conversions

Computer memory is made up of bits, and bytes, and ...

Zoom in (a lot!)

Specialized circuits and electronics provide many “spaces,” each of which is capable of holding a value ...
Binary, Decimal conversions

Computer memory is made up of bits, and bytes, and ...

Each value in a “space” can be a 0 or a 1
Computer memory is made up of bits, and bytes, and ...

Each “space” in computer science is called a **bit**, and each bit can be either a 0 or a 1.
8 bits is a byte. Common prefixes such as mega and giga are used in combination with byte, addition to kilo and tera.

- \( \text{kilo} = 10^3 = 1000 \)
- \( \text{mega} = 10^6 = 1000000 \)
- \( \text{giga} = 10^9 = 1000000000 \)
- \( \text{tera} = 10^{12} = 1000000000000 \)

These are for the decimal (base 10) system.

Q: Does kilobyte mean 1000 bytes?
8 bits is a **byte**. Common prefixes such as **mega** and **giga** are used in combination with byte, addition to **kilo** and **tera**.

<table>
<thead>
<tr>
<th>kilo</th>
<th>mega</th>
<th>giga</th>
<th>tera</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^3 = 1000$</td>
<td>$10^6 = 1000000$</td>
<td>$10^9 = 1000000000$</td>
<td>$10^{12} = 1000000000000$</td>
</tr>
</tbody>
</table>

These are for the decimal (base 10) system.

In computer science, everything is done in base 2 (0 or 1), thus the kilo, mega, giga, and tera mean slightly different things when used in combination with “byte.”
You don’t have to memorize these ... just appreciate that if your computer has 8 gigabytes of memory, that’s a LOT of bytes, and even more bits!

8,589,934,592 bytes
68,719,476,736 bits

kilo = $10^3 = 1000$

megabyte = kilobyte = $2^{10} =$
mega = $10^6 = 1000000$
giga = $10^9 = 1000000000$
tera = $10^{12} = 1000000000000$

terabyte =

Q: How many bytes?
Binary, Decimal conversions

You don’t have to memorize these ... just appreciate that if your computer has 8 gigabytes of memory, that’s a LOT of bytes, and even more bits!

8,589,934,592 bytes
68,719,476,736 bits

kilo = $10^3 = 1000$
kilo byte = $2^{10} = 1,024$ bytes
mega = $10^6 = 1,000,000$
mega byte =
giga = $10^9 = 1,000,000,000$
giga byte =
tera = $10^{12} = 1,000,000,000,000$
tera byte =

Q: What are the base and exponents for megabyte, gigabyte, and terabyte
Binary, Decimal conversions

You don’t have to memorize these ... just appreciate that if your computer has 8 gigabytes of memory, that’s a LOT of bytes, and even more bits!

8,589,934,592 bytes
68,719,476,736 bits

kilo = $10^3 = 1000$
kilo byte = $2^{10} = 1024$ bytes

mega = $10^6 = 1000000$
megabyte = $2^{20} = 1048576$ bytes

giga = $10^9 = 1000000000$
gigabyte = $2^{30} = 1073741824$ bytes

tera = $10^{12} = 1000000000000$
terabyte = $2^{40} = 1099511627776$ bytes
Binary, Decimal conversions

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

First, about base-10 versus base-2: All that this means is that base-10 can use 10 different digits (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9) to “count”, while base-2, or binary, can only use 2 digits (0 or 1).
That still doesn’t answer “how” data is stored in memory.

So let’s start with a simple question:

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

First, about base-10 versus base-2: All that this means is that base-10 can use 10 different digits (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9) to “count”, while base-2, or binary, can only use 2 digits (0 or 1).

Because computers only use 1s and 0s, there MUST be a way for a computer to be able to store any base-10 digit using only 1s and 0s.
Binary, Decimal conversions

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

Insight: every base-10 number can be written as a summation of 2s raised to some power
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

\[ 5 = 2^2 + 2^0 \]
That still doesn’t answer “how” data is stored in memory.

So let’s start with a simple question.

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

**Insight:** every base-10 number can be written as a summation of 2s raised to some power.

\[
5 = 2^2 + 2^0 = 4 + 1
\]
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

5 = 2^2 + 2^0
   = 4 + 1

This is a mathematical “rule” ... any base number raised to a 0 power is 1
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

5 = 2^2 + 2^0
= 4 + 1
= 5

This is a mathematical “rule” ... any base number raised to a 0 power is 1
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

Q: What is the base-10 number 6 written using only powers of 2?
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

Q: What is the base-10 number 6 written using only powers of 2?

$$6 = 2^2 + 2^1$$

$$= 4 + 2$$

$$= 6$$
That still doesn’t answer “how” data is stored in memory

Insight: every base-10 number can be written as a summation of 2s raised to some power

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

Q: How does this insight help us to “store” base-10 numbers as 0s and 1s?
Binary, Decimal conversions

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

Insight: every base-10 number can be written as a summation of 2s raised to some power

A byte is made up of 8 bits
Each “bit” in a byte refers to a specific power of 2

Q: What does that mean?
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power.

That still doesn’t answer “how” data is stored in memory.

So let’s start with a simple question.

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

\[
2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0
\]
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

```
0 1 0 0 0 1 0 1
```

\[ 2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \]

Q: How is then a number “stored” as a sequence of 0s and 1s?
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

If there is a 1 in a bit space for a byte, then you INCLUDE that power of 2 in a summation

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^7$</td>
<td>$2^6$</td>
<td>$2^5$</td>
<td>$2^4$</td>
<td>$2^3$</td>
<td>$2^2$</td>
<td>$2^1$</td>
<td>$2^0$</td>
</tr>
</tbody>
</table>
Insight: every base-10 number can be written as a summation of 2s raised to some power

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

Q: What is 2 to the zero power?
That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

0 1 0 0 0 1 0 1

2^7 2^6 2^5 2^4 2^3 2^2 2^1

2^0

1

Insight: every base-10 number can be written as a summation of 2s raised to some power
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

Q: What is 2 to the power 2?
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

Q: What is 2 to the power 6?
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power.

That still doesn’t answer “how” data is stored in memory.

So let’s start with a simple question:

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2^7</td>
<td>2^6</td>
<td>2^5</td>
<td>2^4</td>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

\[
2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 64 + 4 + 1
\]

Q: And if you sum these up?
Binary, Decimal conversions

Insight: every base-10 number can be written as a summation of 2s raised to some power

That still doesn’t answer “how” data is stored in memory

So let’s start with a simple question

Q: How is the number 5 in base-10 notation stored using just 0s and 1s?

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^7$</td>
<td>$2^6$</td>
<td>$2^5$</td>
<td>$2^4$</td>
<td>$2^3$</td>
<td>$2^2$</td>
<td>$2^1$</td>
<td>$2^0$</td>
</tr>
</tbody>
</table>

\[
64 + 4 + 1 = 69
\]

Therefore, $01000101$ is the binary encoding of the number 69.
Q: What is the base-10 equivalent of the binary number 000001101?
Q: What is the base-10 equivalent of the binary number 000001101?

\[
\begin{align*}
0 &= 2^0 \\
1 &= 2^2 \\
1 &= 2^3 \\
0 &= 2^4 \\
1 &= 2^5
\end{align*}
\]

\[
= 8 + 4 + 1
= 13
\]
Q: What is the binary equivalent of the number 44?

(on the board explanation)
That’s the main idea

A trickier question:
Q: How are “letters” stored in memory?
That’s the main idea

A trickier question:
Q: How are “letters” stored in memory?

There is an intermediate conversion process that happens BEFORE anything is stored in memory.

The conversion process is based on the ASCII table.
That’s the main idea

A trickier question:
Q: How are “letters” stored in memory?

To store the letter “R”, the binary equivalent of the number 82 is stored in memory
Q: What is the binary equivalent of the year of your birth?
Reassignment and equality
Debugging and the Coding Process
Conditionals