**Objectives**
- Review Linux, algorithms
- Programming in Python
  - Data types
  - Expressions
  - Variables
  - Arithmetic
- Broader Issues

**Review: Linux File System**
HOME directory (~)
```
/  home  students  yourname
  ...  ...  labs  public_html
```
- is a shortname for your home directory, i.e., short for /home/students/yourname
- What is the *syntax* for the copy command?
- How would you copy `practice.py` to your `public_html` directory?

**Review**
- What is an algorithm?
- What are the parts of an algorithm?
- Why do we need programming languages?
- What are some properties of programming languages?

**Totally Inappropriate Example of an Algorithm**

**Parts of an Algorithm**
- Input, Output
  - Primitive operations
    - What data you have, what you can do to the data
  - Naming
    - Identify things we're using
  - Sequence of operations
  - Conditionals
    - Handle special cases
  - Repetition/Loops
  - Subroutines
    - Call, reuse similar techniques

**Primitive Data Types**
- Primitive data types represent data
  - In PB&J example, our data had types slice of bread, PB jar, jelly jar, etc.
- Python provides some basic or primitive data types
- Broadly, the categories of primitive types are
  - Numeric
  - Boolean
  - Strings
### Numeric Primitive Types

<table>
<thead>
<tr>
<th>Python Data Type</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int</code></td>
<td>Plain integers (32-bit precision)</td>
<td>-214, -2, 0, 2, 100</td>
</tr>
<tr>
<td><code>float</code></td>
<td>Real numbers</td>
<td>0.001, -1.234, 1000.1, 0.00, 2.45</td>
</tr>
<tr>
<td><code>long</code></td>
<td>Bigger integers (neg or pos, precision limited by computer memory)</td>
<td>2147483648L</td>
</tr>
<tr>
<td><code>complex</code></td>
<td>Imaginary numbers (have real and imaginary part)</td>
<td>1j * 1j --&gt; (-1+0j)</td>
</tr>
</tbody>
</table>

---

### How big (or small or precise) can we get?

- We cannot represent all values
- Problem: Computer has a finite capacity
  - The computer only has so much memory that it can devote to one value.
  - Eventually, reach a cutoff
    - Limits size of value
    - Limits precision of value

In reality, computers represent data in binary, using only 0s and 1s.

![Image of binary representation of PI](image.png)

PI has more decimals, but we're out of space!

---

### Strings: `str`

- Indicated by double quotes "" or single quotes ‘’
- Treat what is in the "" or ‘’ literally
  - Known as string literals
- Examples:
  - "Hello, world!"
  - 'c'
  - "That is Buddy's dog."

Can have single quote only inside double quotes*

* Exception later

---

### Booleans: `bool`

- 2 values
  - `True`
  - `False`

More on these later...

---

### What is the value’s type?

<table>
<thead>
<tr>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td><code>int</code></td>
</tr>
<tr>
<td>-0.01</td>
<td></td>
</tr>
<tr>
<td>4+6j</td>
<td></td>
</tr>
<tr>
<td>&quot;int&quot;</td>
<td></td>
</tr>
<tr>
<td>4047583648L</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td></td>
</tr>
<tr>
<td>'false'</td>
<td></td>
</tr>
</tbody>
</table>

---

### Introduction to Variables

- Variables save data/information
  - Example: first slice of bread or knife #1
  - Type of data the variable holds can be any of primitive data types as well as other data types we'll learn about later
- Variables have names, called identifiers
**Variable Names/Identifiers**

- A variable name (identifier) can be any one word that:
  - Consists of letters, numbers, or _
  - Cannot start with a number
  - Cannot be a Python reserved word
    - like `for`, `while`, `def` etc.
  - Python is case-sensitive:
    - `change` isn't the same as `Change`

**Variable Name Conventions**

- **Variables** start with lowercase letter
- **Constants** (values that won't change) are in all capitals

  - Example: Variable for the current year
    - `currentYear`
    - `current_year`
    - `current_year`
    - `CURRENT_YEAR`

---

**Naming Variables**

- Naming is important
  - Helps you **remember** what the variable represents
  - Easier for others to **understand** your program

- Examples:

<table>
<thead>
<tr>
<th>Info Represented</th>
<th>Good Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A person's first name</td>
<td>firstName, first_name</td>
</tr>
<tr>
<td>Radius of a circle</td>
<td>radius</td>
</tr>
<tr>
<td>If someone is employed or not</td>
<td>isEmployed</td>
</tr>
</tbody>
</table>

What are the **types** of each of these variables?

---

**Modeling Information**

- How would you **model** this information?
- What data type best represents the info?

<table>
<thead>
<tr>
<th>Info Represented</th>
<th>Data Type</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A person's salary</td>
<td>Sales tax</td>
<td></td>
</tr>
<tr>
<td>Sales tax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>If item is taxable</td>
<td>Course name</td>
<td></td>
</tr>
<tr>
<td>Course name</td>
<td>Gender</td>
<td></td>
</tr>
<tr>
<td>Gender</td>
<td>Middle initial</td>
<td></td>
</tr>
<tr>
<td>Middle initial</td>
<td>Graduation Year</td>
<td></td>
</tr>
</tbody>
</table>

---

**Assignment Statements**

- Variables can be given any value using the "=" sign
  - Syntax: `<variable> = <expression>`
  - Semantics: `<variable>` is set to value of `<expression>`

- After a variable is set to a value, the variable is said to be **initialized**

- Examples:
  
  ```
  currentYear = 2008
  my_num = 3.4
  option = 'q'
  ```

  These aren't equations! Read "=" as "gets"

- Statements execute in order, from top to bottom
- Value of `x` does not change because of second assignment statement
• Statements execute in order, from top to bottom
• Value of \( x \) does not change because of second assignment statement

### Variables: The Rules

- Only the variable(s) to left of the \( = \) change
  - We’ll usually only have one variable on the left
- Initialize a variable **before** using it on the right-hand side (rhs) of a statement

### Literals

- Pieces of data that are not variables are called **literals**
  - We’ve been using these a lot already
- Examples:
  - 4
  - 3.2
  - ‘q’
  - “books”

### Numeric Arithmetic Operations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
</tr>
<tr>
<td>%</td>
<td>Remainder (“mod”)</td>
</tr>
<tr>
<td>**</td>
<td>Exponentiation (power)</td>
</tr>
</tbody>
</table>

### Arithmetic & Assignment

- You can use the assignment operator \( (=) \) and arithmetic operators to do calculations
  - First, calculate right hand side
  - Then, assign value to variable
- Remember your order of operations! (PEMDAS)
- Examples:
  - \( x = 4 + 3 \times 10 \)
  - \( y = 3.0 / 2.0 \)
  - \( z = x + y \)
  - The right-hand sides are expressions, just like in math.
  - For 3rd statement, need to “lookup” values of \( x \) and \( y \)
Examples:

- \( x = 4 + 3 \times 10 \)
- \( y = 3.0 / 2.0 \)
- \( z = x + y \)

For 3rd statement, need to “lookup” values of \( x \) and \( y \)

- Note that \( x \) and \( y \) do not change because of \( z \)’s assignment statement

After executing the following statements, what are the values of each variable?

- \( x = 5 \)
- \( y = -1 + x \)
- \( z = x + y \)
- \( y = 2 \)
- \( x = -7 \)

How can we verify our answers?

After executing the following statements, what are the values of each variable?

- \( a = 5 \)
- \( y = a + -1 \times a \)
- \( z = a + y / 2 \)
- \( a = a + 3 \)
- \( y = (7 + x) \times z \)
- \( x = z \times 2 \)

Extra Credit Opportunity

- 10 points applied to Lab grade
- Attend a CS talk, all in Parmly 405
  - Mon, Jan 12, D period
  - Thurs, Jan 15, 3:30 p.m.
    - Mark Liffiton, “Satisfying Constraints, and What To Do When You Can’t”
  - Fri, Jan 23, 4 p.m.
    - Joshua Stough, “Appearance Models for Medical Image Segmentation”
- Post summary on Sakai, following CS Issues write up

Testing for Tech Literacy

- Groups
  - Juniors: Mike, Kevin, Ben, Russ
  - Sophomores: Camille, Charles, Carrie, Peter
  - FirstYears: Michelle, Aaron, David, Chen
  - Seniors: Craig, Dylan, Thomas, Mallory

Groups
- Juniors: Mike, Kevin, Ben, Russ
- Sophomores: Camille, Charles, Carrie, Peter
- FirstYears: Michelle, Aaron, David, Chen
- Seniors: Craig, Dylan, Thomas, Mallory
Broader CS Issues
- Good summaries!
  - Good English, complete sentences
- Good, thoughtful questions

Mechanics details
- Follow instructions on “CS Issues” about what summary should contain
- Should be able to edit your own posts
- Still some Word characters
  - View your post after you write it
  - Fix as necessary

Tech Literacy
- Each group come to a decision on: Should technical skills be taught in schools?
  - If so, what skills should be taught at 4th, 8th, and 12th grade?
  - Rank the skills within your group
  - If not, why?
- What do you think of the statement “What is assessed is what is taught”?
- Will accessing technology skills affect the haves and have not schools differently?
- What is the difference between “technology education” and “computer science”?

My Tech Literacy Notes
- Your lab: “These students built their pages from scratch, writing pure HTML in a text file.”
- After taking this course, I’m not sure how you’d do on those assessment tests
  - E.g., you won’t have advanced knowledge of Microsoft Office
- You will be better, more logical thinkers
  - Better problem solvers
  - Toward efficiency experts

Extra Credit Opportunities
- Read an article that relates to CS
- Summarize it on the forum under “Extra Credit”
  - 5 pts extra credit on lab grade