### What Are Algorithms?

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Step-by-step instructions</td>
<td></td>
</tr>
<tr>
<td>• Not specific to a language</td>
<td></td>
</tr>
<tr>
<td>• Could be a cooking recipe</td>
<td></td>
</tr>
<tr>
<td>• <strong>Outline</strong> for a program</td>
<td></td>
</tr>
<tr>
<td>• Program for an algorithm</td>
<td></td>
</tr>
<tr>
<td>• In a specific language</td>
<td></td>
</tr>
<tr>
<td>• What we often call coding</td>
<td></td>
</tr>
<tr>
<td>• The <strong>filled in</strong> outline</td>
<td></td>
</tr>
</tbody>
</table>

- Good programmers can separate the two
  - Work on the algorithm first
  - Implement in language second
- Why approach strings as search-cut-glue

### Difficulties With Programming

- **Syntax Errors**
  - Python can’t understand you
  - **Examples:**
    - Forgetting a colon
    - Not closing a parens
  - Common with beginners
  - But can quickly train out

- **Conceptual Errors**
  - Does what you say, not mean
  - **Examples:**
    - Forgot last char in slice
    - Used the wrong argument
  - Happens to everyone
  - Large part of CS training

- Proper algorithm design reduces conceptual errors

### Testing First Strategy

- **Write the Tests First**
  - Could be script or written by hand
- **Take Small Steps**
  - Do a little at a time; make use of **placeholders**
- **Intersperse Programming and Testing**
  - When you finish a step, test it immediately
- **Separate Concerns**
  - Do not move to a new step until current is done

### Using Placeholders in Design

- **Strategy:** fill in definition a little at a time
- **We start with a function** *stub*
  - Function that can be called but is unfinished
  - Allows us to test while still working (later)
- **All stubs must have a function header**
  - But the definition body might be “empty”
  - Certainly is when you get started

### A Function Stub

```python
def last_name_first(s):
    """Returns: copy of s in form 'last-name, first-name'
    Precondition: s is in form 'first-name last-name'
    with one blank between the two names""
    pass
```

Now pass is a note that is unfinished. Can leave it there until work is done.

### Outlining Your Approach

```python
def last_name_first(s):
    """Returns: copy of s in form 'last-name, first-name'
    Precondition: s is in form 'first-name last-name'
    with one blank between the two names""
    # Find the space between the two names
    # Cut out the first name
    # Cut out the last name
    # Glue them together with a comma
```

Psuedocode
What is the Challenge?

• Pseudocode must correspond to Python
  - Preferably implementable in one line
  - **Unhelpful:** # Return the correct answer
• So what can we do?
  - Depends on the types involved
  - Different types have different operations
  - You should memorize important operations
  - Use these as building blocks

---

Working with Helpers

• Suppose you are unsure of a step
  - You maybe have an idea for pseudocode
  - But not sure if it easily converts to Python
• But you can **specify** what you want
  - Specification means a **new function**!
  - Create a specification stub for that function
  - Put a call to it in the original function
• Now can **lazily** implement that function

---

A Word of Warning

• **Do not go overboard** with this technique
  - Do not want a lot of one line functions
  - Can make code harder to read in extreme
• Do it if the **code is too long**
  - I personally have a one page rule
  - If more than that, turn part into a function
• Do it if you are **repeating yourself a lot**
  - If you see the same code over and over
  - Replace that code with a single function call

---

Stubbed Returns for Incremental Testing

```python
def last_name_first(s):
    """Returns: copy of s in form 'last-name, 'first-name'
Precondition: s is in form 'first-name last-name' with one blank between the two names"
end_first = introcs.find_str(s,' ')
first = s[end_first]
# Cut out the last name
# Glue them together with a comma
return first  # Not the final answer
```

---

Example: last_name_first

```python
def last_name_first(s):
    """Returns: copy of s in form 'last-name, 'first-name'
Precondition: s is in form 'first-name last-name' with one blank between names"
end_first = introcs.find_str(s,' ')
first = s[end_first]
# Cut out the last name
# Glue them together with a comma
return first  # Stub
```

---

Exercise: Anglicizing an Integer

```python
def anglicize(n):
    """Returns: the anglicization of int n.
Precondition: 0 < n < 1,000,000"
pass # ???
```