Making our program more robust
Here is our current program

```python
n = int(raw_input("Enter a positive integer:"))
suffix = ""
originalN = n
while n > 0:
    suffix = str(n%2) + suffix
    n = n/2
print "The binary equivalent of", originalN, "is", suffix
```
Making the program more robust

- What if the user types in a negative integer or 0? Or a real number? Or some non-numeric string, (e.g., “hello”)?

- We will only discuss the negative integer or 0 situation now.

- Later when we discuss exceptions and how to handle them, we’ll return to this program.
**Types of errors**

- **Syntax error**
  
  Syntax refers to the structure of the program. (e.g., English sentences start with a capital letter)

**Examples:**

```python
while x < 10:
    x = x + 1
n = int(input())
print(n)
```
Types of errors

- **Run-time errors (or exceptions)**
  This is an error that occurs during the running of the program and is typically caused by the user not anticipating a certain behavior of their program.

**Example:**

```python
n = int(raw_input("Enter a number:"))
print n + 5
```

What if the user inputs "hello"?
Types of errors

- **Semantic errors**
  The program may not produce an error message when executed, but it may not do what we expect it to do.

  **Example:**
  In an earlier version of our program:
  ```python
  print "The binary equivalent of", n, "is", suffix
  ```
  We forgot that n would have changed to 0 at this point.
The case of non-positive integers

- What does the program currently do, if the user inputs a negative integer or 0?

- We could instead try to print an informative message.

- We will use the if-else statement for that.
Simple `if` statement

```
if boolean expression:
    Line 2
    Line 3
    Line 4
```

- If boolean expression is true:
  Line 1, Line 2, Line 3, Line 4.
- Otherwise: Line 1, Line 4.
**if-else statement**

- If boolean expression is true:
  - Line 1, Line 2, Line 3, Line 5

- Otherwise: Line 1, Line 4, Line 5
Dealing with negative integer input

- If \( n \leq 0 \), print out an appropriate message and do nothing else.
- Else, continue to do what the program is currently doing.
Our Final First Program

```python
n = int(input("Enter a positive integer:"))
if n <= 0:
    print "Enter a positive integer next time. Bye!"
else:
    suffix = ""  
    originalN = n 
    while n > 0:
        suffix = str(n%2) + suffix 
        n = n/2
    print "The binary equivalent of", originalN, "is", suffix
```
Primality Testing
Given a positive integer (> 1), determine whether it is a prime number or not.

Examples:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>prime</td>
</tr>
<tr>
<td>2001</td>
<td>composite</td>
</tr>
<tr>
<td>987654321</td>
<td>composite</td>
</tr>
</tbody>
</table>