Understanding the First Python Program
Our first program

```python
n = int(raw_input("Enter a positive integer:"))
while n > 0:
    print n % 2
    n = n/2
```

n = int(raw_input("Enter a positive integer:"))

Assignment statement
• = is the assignment operator

• n is a variable

• The stuff on the right hand side is an expression that gets evaluated and its value gets assigned to the variable n
Examples of assignment statements

- \( n = 9 \)
- \( n = n/2 \)
  (Assignment operator is not algebraic equality)
- \( n = n + 1 \)
  (A commonly used assignment statement)
- \( n = \text{math.sqrt}(100) \)
  (Can be used after importing the math module)
- \( n = \text{raw_input}("Enter a number:") \)
The `raw_input` function

`raw_input(prompt)`

- This function is a built-in Python function and is always available.

- The `prompt` is written to output and then the function reads a line from input.
raw_input evaluates to a string

Try this code snippet. What happens?

```python
x = raw_input("Enter a number:'")
x = x + 1
```

What the user types is read in as a string, the expression on the right hand side evaluates to a string and x gets assigned a string.
Data types in Python

- Every object (e.g., constants, variables) in Python has a type

- An object’s type determines what operations can be performed on that object.

- Use the Python built-in function `type` to determine an object’s data type.
Data types in Python

- **Examples:**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Enter a positive integer”</td>
<td>string</td>
</tr>
<tr>
<td>2</td>
<td>Integer</td>
</tr>
</tbody>
</table>

- Python has many built-in data types. For now we will work with three types:

  - integer: int
  - string: str
  - floating point: float
The type of a variable is the type of what it was most recently assigned.

**Example:**

```python
x = 15
print(type(x))  # int
x = x*1.0
print(type(x))  # float
```

This ability of the same variable to have different types within a program is called *dynamic typing*. 
Operators and data types

- The meaning of *operators* (e.g., +, /) depends on the data types they are operating on.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9/2</td>
<td>4</td>
</tr>
<tr>
<td>9.0/2</td>
<td>4.5</td>
</tr>
<tr>
<td>9/2.0</td>
<td>4.5</td>
</tr>
<tr>
<td>5 + 1</td>
<td>6</td>
</tr>
<tr>
<td>5 + 1.0</td>
<td>6.0</td>
</tr>
<tr>
<td>“hello,”+” friend”</td>
<td>“hello, friend”</td>
</tr>
</tbody>
</table>
Conversions between data types

- Python provides built-in functions for converting between data types.

**Examples:**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>int(“320”)</td>
<td>320</td>
</tr>
<tr>
<td>float(“320”)</td>
<td>320.0</td>
</tr>
<tr>
<td>str(134)</td>
<td>“134”</td>
</tr>
</tbody>
</table>
n = int(raw_input("Enter a positive integer:"))

1. raw_input prints the prompt and reads a line of the user’s input as a string.

2. This string gets converted to an integer by the function int.

3. This integer gets assigned to the variable n.
Understanding the \texttt{while} statement

\begin{verbatim}
while n > 0:
    body of while
\end{verbatim}

- Usually the execution of a program \textit{flows} from Line 1 to Line 2 to Line 3...

- All programming languages provide statements that affect this flow. These are called \textit{flow control} statements.

- The \texttt{while} statement is a flow control statement.
How do **while** statements affect program flow?

Flow
Line 1, bool expr, Line 2, Line 3, bool expr, Line 2, Line 3, ...
bool expr
Line 4