Improving our program

JAN 30TH 2013

Improving the output

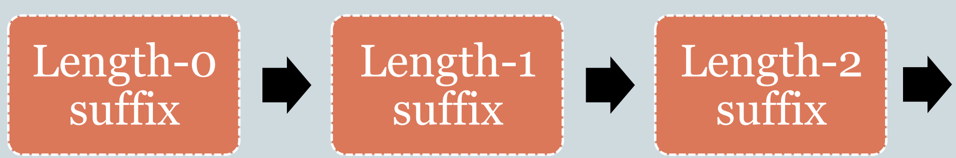
- How can we put together the bits we generate, in the correct order, to construct the binary equivalent?

- **String concatenation**!

Expression	Value
“0” + “1001”	“01001”
“1” + “1001”	“11001”

Algorithmic idea

- After *i* iterations of the while loop we have generated the right most *i* bits of our answer.
- Call this the *length-i suffix*.
- We want to maintain a string that grows as:



Example

- Input is 39.

Output	Suffix
1	“”
1	“1”
1	“11”
0	“111”
0	“0111”
1	“00111”
	“100111”

Improved program

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

Here is another improvement to the output

```
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.

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:

```
while x < 10
        x =x + 1
```

```
n = int(raw_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:

```
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:

```
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

```
Line 1
if boolean expression:
        Line 2
        Line 3
Line 4
```

- If boolean expression is true:

  Line 1, Line2, Line 3, Line 4.
- Otherwise: Line 1, Line 4.

if-else statement

```
Line 1
if boolean expression:
        Line 2
        Line 3
else:
        Line 4
Line 5
```

- 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

One possible approach:

- If n <= 0, print out an appropriate message and do nothing else.
- Else, continue to do what the program is currently doing.

Our Final First Program

```
n = int(raw_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
```