Understanding our first program

JAN 23RD 2012

Understanding the input statement

```
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 = math.sqrt(100)`

(Can be used after importing the math module)

- `n = 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?

```
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:

Constant	Type
"Enter a positive integer"	string
2	Integer

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

integer	int
string	str
floating point	float

Type of a variable

- The type of a variable is the type of what it was most recently assigned.

Example:

```
x = 15
type(x)                 int
x = x*1.0
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.

Expression	Value
9/2	4
9.0/2	4.5
9/2.0	4.5
5 + 1	6
5 + 1.0	6.0
"hello,"+" friend"	"hello, friend"

Conversions between data types

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

- **Examples**:

Expression	Value
int("320")	320
float("320")	320.0
str(134)	"134"

Last slide on the first line

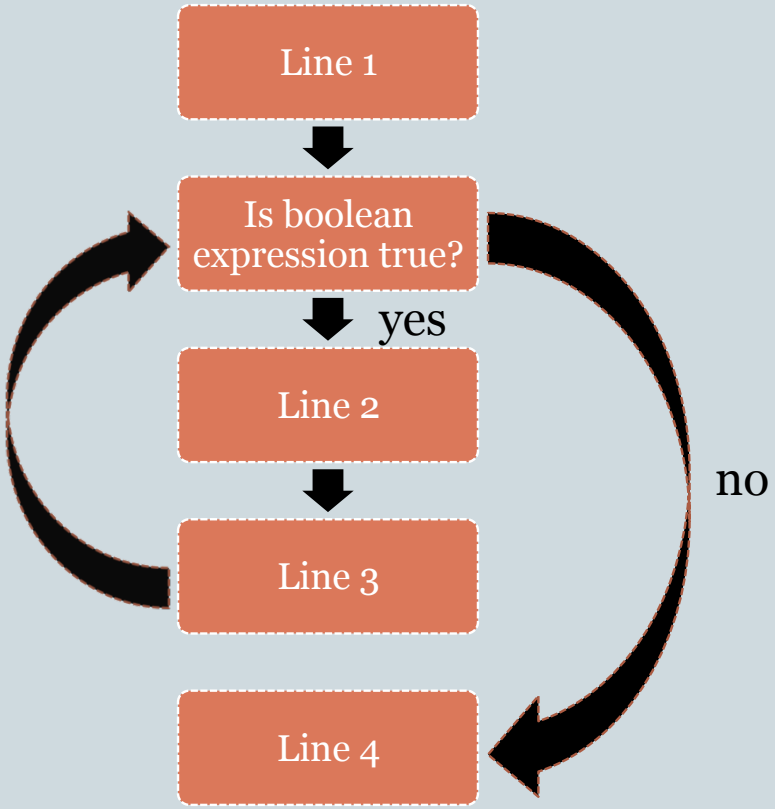
```
n = int(raw_input("Enter a positive integer:"))
```

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`.

How do while statements affect program flow?

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

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



Body of while loop

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

- Lines 2 and 3 form the *body* of the while loop
- Python uses indentation to identify the lines following the while statement that constitute the body of the while loop.

Boolean expressions

- Python has a type called `bool`
- The constants in this type are `True` and `False`.
  (Not `true` and `false`!)
- The comparison operators:

  `<` `>` `<=` `>=` `!=` `==`

  can be used to construct *boolean expressions*, i.e., expressions that evaluate to `True` or `False`.

Boolean expressions: examples

- Suppose x has the value 10

Expression	Value
x < 10	False
x != 100	True
x <= 10	True
x > -10	True
x >= 11	False

A silly while loop example

```
n = int(raw_input("Enter a positive integer:"))
while n != 0:
        n = n - 2
```

- What happens when input is 8?
- What happens when the input is 9?

The biggest danger with while loops is that they may run forever.