

# ASSIGNMENT, LOOPS, AND BASIC TYPES

# Outline

---

- Variables and assignments
- Definite loops
- Basic types: numbers (int and float)
- Math library
- Accumulator problem

# Variables and Assignments

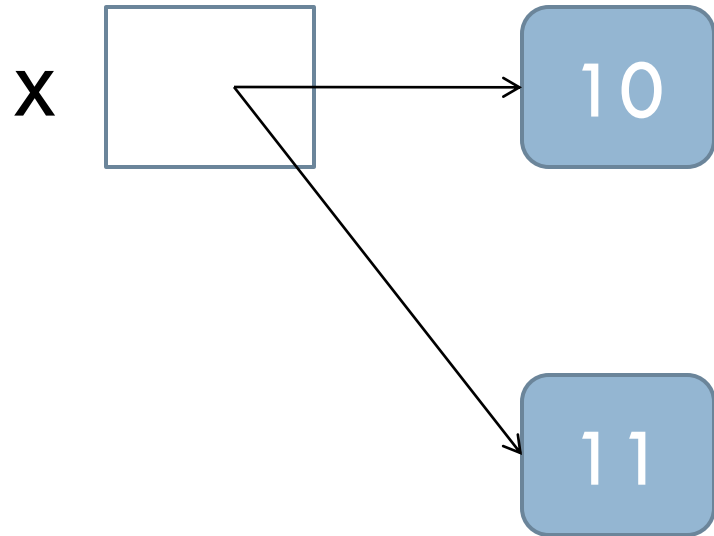
## □ Variable

- Identifier that stores a value
- A value must be *assigned* to the variable
- $\langle \text{variable} \rangle = \langle \text{expr} \rangle$  (syntax)

## □ Assignment

- Process of giving a value to a variable
- Uses = (equal sign)
  - $x = 0.25$
  - $x = 3.9 * x * (1 - x)$

# Variables as sticky notes



**x = 10**

**x = x + 1**

# Assignment Statements

- Simple assignments
  - ▣ `<variable> = <expr>`
- Input assignments
  - ▣ `<variable> = input(<prompt>)`
    - `temp = input("Enter high temperature for today")`
- Simultaneous assignments
  - ▣ `<var>, <var>, ..., <var> = <expr>, <expr>, ..., <expr>`
    - `sum, diff = x + y, x - y`

# Sequences

- A list of things
- For example:
  - [2, 3, 5, 7]
  - ["My", "dog", "has", "fleas"]
- Generated by the range function
  - range(<expr>)
  - range(<expr>, <expr>)
  - range(<expr>, <expr>, <expr>)

# Definite loops

- Definition

- **Control structure** for executing a portion of a program multiple times
- **Definite**: Python **knows** how many times to **iterate** body of the loop

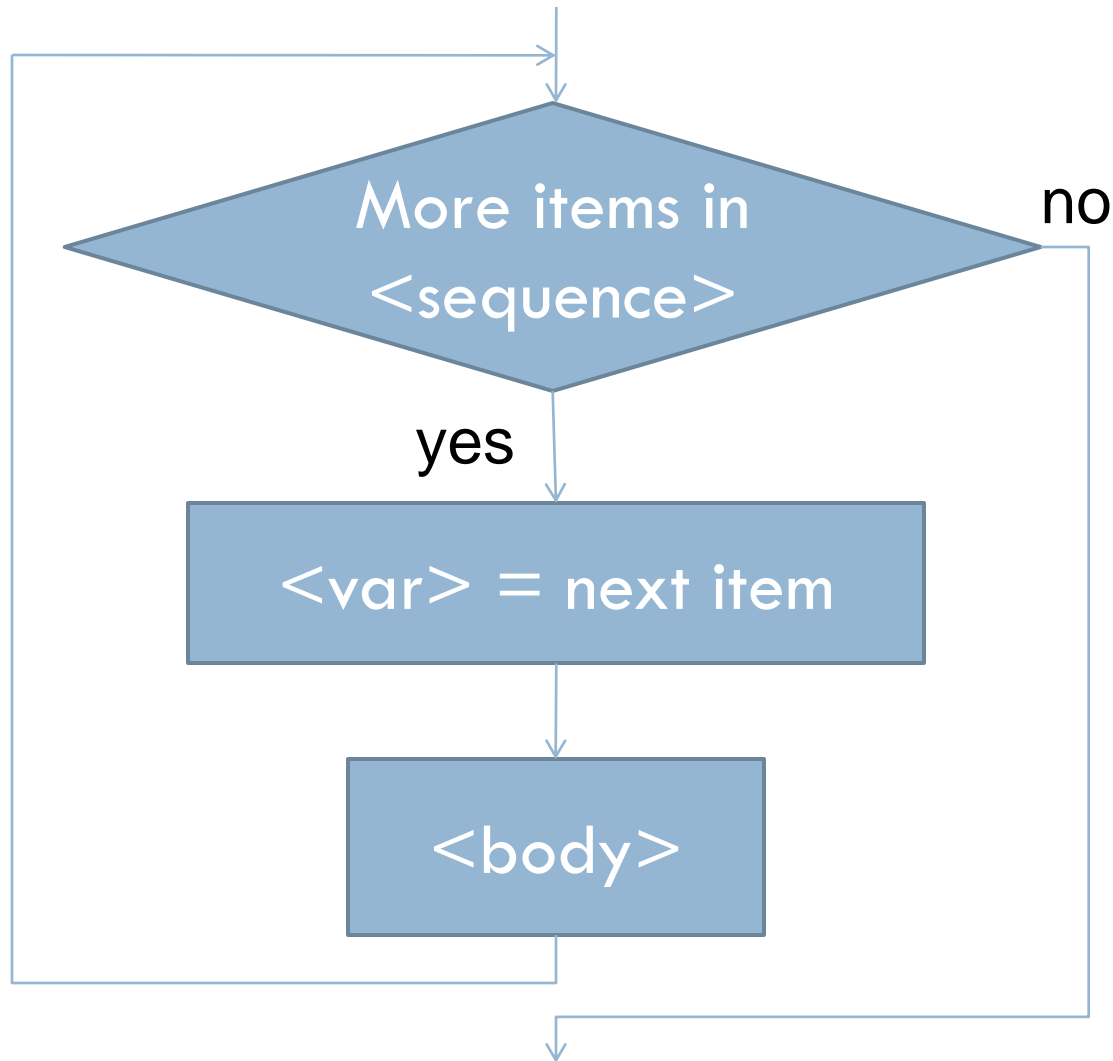
- Syntax:

```
for <var> in <sequence>:  
    <body>
```

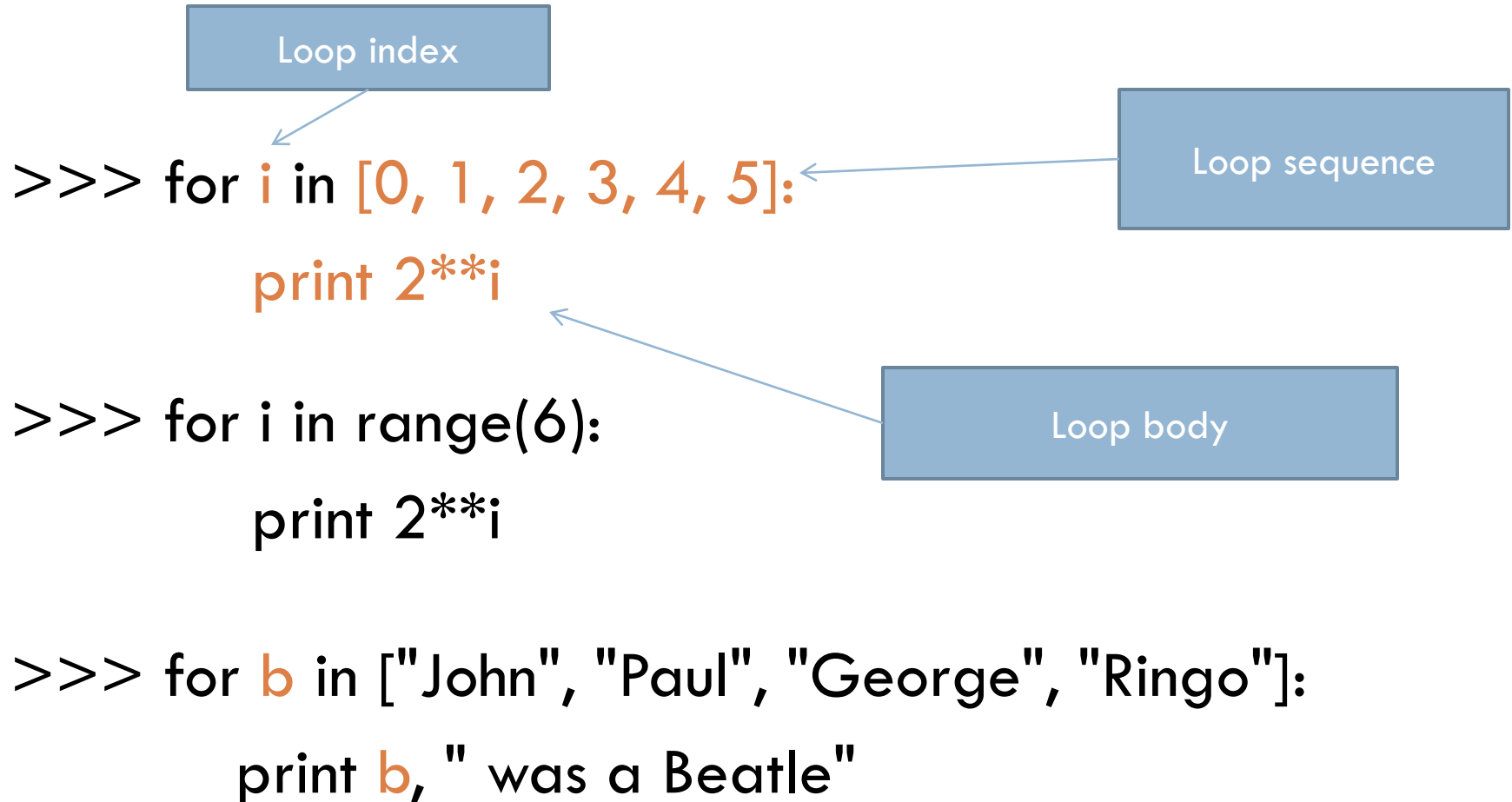
# Counted Loops

- A definite loop whose sequence can be generated with `range(...)`
- Example counted loop:
  - ▣ `for x in range(2,4):`  
    `print x`
- Example definite loop that isn't a counted loop:
  - ▣ `for p in [2, 3, 5, 7, 11]:`  
    `print p, "is prime"`

# Flowchart for a for loop



# Examples using loops



# Data types

## □ *Data*

- Information stored and manipulated on a computer
- Different kinds of data will be stored and manipulated in different ways

## □ *Data types*

- A particular way of interpreting bits
- Determine possible values an item can have
- Determine operations supported on items

# Numeric data types

- From Angel copy the content of the following file and place it in a new Python window:

Lessons → Modules to Download in Class →  
Session 3 → change.py

- Save as myChange.py
  
- Numeric types
  - Whole numbers    int
  - Fractional numbers    float

# Finding the Type of Data

- Built-in function `type(<expr>)` returns the data type of any value
- Find the types of: 3, 3.0, -32, 64.0, “Shrubbery”
- Why do we need different numerical types?
  - ▣ Operations on int are more efficient
    - Compute algorithm for operations on int are simple and fast
  - ▣ Counting requires int
  - ▣ Floats provide approximate values when we need real numbers

# Some Numeric Operations

Operator	Operation
+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation
%	remainder
//	Do integer division (even on floats)

Function	Operation
abs(x)	Absolute value of x
round(x, y)	Round x to y decimal places
int(x)	Convert x to an int data type
float(n)	Convert n to a float data type

# Using Built-in Operations

- Modify your `myChange.py` program so it prints a whole number of dollars and a whole number of cents using Python's built-in numeric operations
  - ▣ Left-half of room, can't use `round`
  - ▣ Right-half of room, must use `round`

# Math library functions

From Angel copy the content of the following file and place it in a new Python window:

Lessons → Modules to Download in Class →  
Session 3 → quadratic.py

Save as `myQuadratic.py`

□ Finds real roots for quadratic equations of the form

$$ax^2 + bx + c = 0$$

□ Solution

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

# More math library functions

Python	Mathematics	English
pi	$\pi$	Approximation of pi
e	e	Approximation of e
sin(x)	sin x	The sine of x
cos(x)	cos x	The cosine of x
tan(x)	tan x	The tangent of x
log(x)	ln x	The natural (base e) log of x
log10(x)	$\log_{10}x$	The base 10 log of x
exp(x)	$e^x$	The exponential of x

# Built-in Help

---

- `dir()`
- `dir(<identifier>)`
- `help(<identifier>)`

# EXPLORING WITH PYTHON



# Pair Programming

- Working in pairs on a single computer
  - ▣ One person, the *driver*, uses the keyboard
  - ▣ The other person, the *navigator*, watches, thinks, and takes notes
- For hard (or new) problems the technique:
  - ▣ Reduces number of errors
  - ▣ Saves time in the long run

# Problem 1

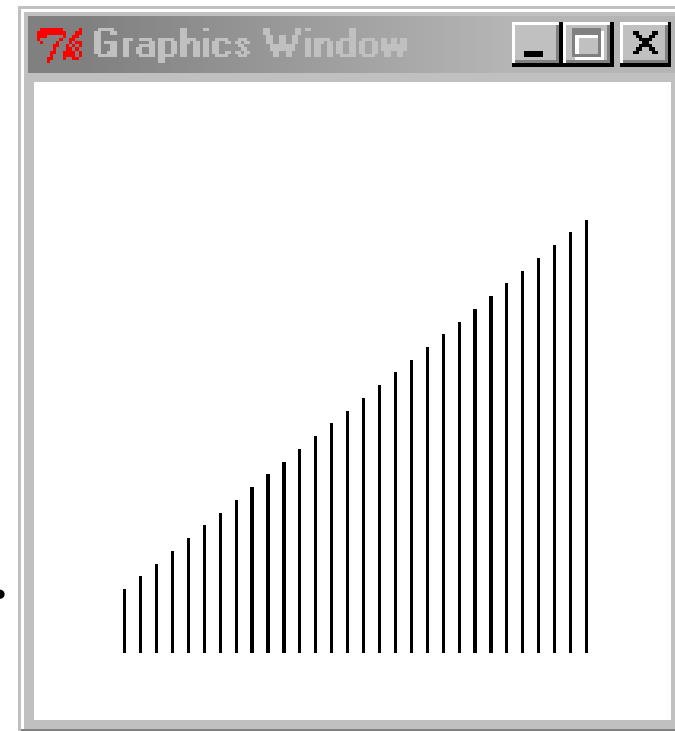
- Suppose you are at food tasting show and are tasting 5 different dishes.
- Sampling the dishes in different orders may affect how good they taste.
- If you want to try out every possible ordering, how many different orders would there be?
  - ▣ That number is the factorial of 5
  - ▣  $n! = n (n - 1) (n - 2) \dots (1)$

# Accumulating results: factorial

- Work in groups of two
  - ▣ Pick one person to drive and the other to navigate
  - ▣ If you have programmed before, try to find a partner who has also done so
- In groups of 2 write a Python program that
  - ▣ Prompts the user for an integer
  - ▣ Calculates the factorial of the integer
    - $n! = n (n - 1) (n - 2) \dots (1)$
  - ▣ Outputs the result to the screen

# Graphics Exercise with loops

- Trade roles with partner—new driver, new navigator
- Write a program that draws a figure like this where the lengths of the lines increase by a constant amount.
- Use your previous graphics program as a model of how to import graphics functions, create a window, etc.
- You may want to use variables to hold current x-coordinate and current line length, and change the values of those variables each time through the loop.



# Submitting your programs

- Meet before next class to finish them
- Driver: email the code to your partner (so each has the program for the open-computer parts of exams)
- Log into to Angel and go to the class's webpage
- Click on the lessons tab then go to the **In-class Exercise Drop Boxes**
- Submit the factorial program in the **Factorial Drop Box**
- Submit the line drawing program in the **Bar Chart Drop Box**