

As you arrive:

1. Start up your computer and plug it in
2. **Log into Angel** and go to CSSE 120
3. Do the **Attendance Widget** – the PIN is on the board
4. Go to the course **Schedule Page**
 - From your *bookmark*, or from the *Lessons* tab in Angel
5. Open the **Slides** for today if you wish

ASSIGNMENT AND LOOPS

Outline (some of chapters 2 and 3)

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

Check out project for today

- Go to SVN Repository view, at bottom of the workbench
 - ▣ If it is not there,
Window → Show View → Other → SVN → SVN Repositories
- Browse SVN Repository view for **03-AssignmentsAndLoops** project
- Right-click it, and choose **Checkout**
 - ▣ Accept options as presented
- Expand the **03-AssignmentsAndLoops** project that appears in Package Explorer (on the left-hand-side)
 - ▣ Browse the modules.
 - ▣ We will start with ***intsAndFloats.py*** (next slide)

Some numeric operations

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation
%	Remainder
//	Integer division

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

Variables

5

```
width = 4
temperature = 98.6
```

```
dogName = "fido"
lost = [4, 8, 15, 16, 23, 42]
```

```
triangleArea = width * height / 2
xyPoint = (r * cos(theta), r * sin(theta))
```

Variables are *identifiers* that refer to *values* stored in *memory*. Case matters – variables `width` and `Width` are independent of each other!

Values can be integers, floating point numbers, strings, lists, and more.

Expressions are built from variables, literals and function calls, and can be *evaluated*.

`<variable> = <expr>`



Variables and assignments

6

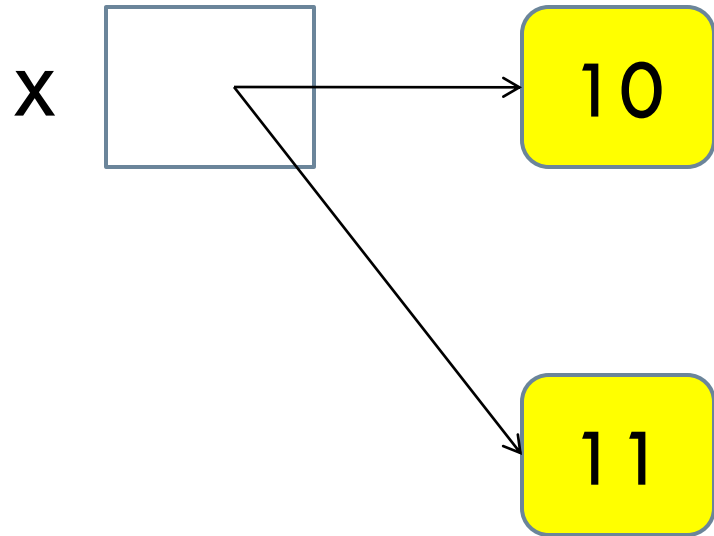
Assignment gives a variable a value, using the `<variable> = <value>` notation. Read it as “gets” or “becomes”. The right-hand-side is evaluated. The name `<variable>` on the left-hand-side then refers (points to) that `<value>`.

```
x = 0.25
```

Statements like the following, while terrible mathematics, are perfectly sensible in software development. The first one, for example, is read as “*x becomes 3.9 times x was plus times 1 minus what x was.*”

```
x = 3.9 * x * (1 - x)
interestRate = interestRate * 1.5
```

Variables as sticky notes



$x = 10$

$x = x + 1$

Assignment statements

1. Simple assignments

- ▣ `<variable> = <expr>`

2. Input assignments

- ▣ `<variable> = input(<prompt>)`

- ▣ `temp = input("Enter high temperature for today")`

3. Compound assignments

- ▣ `<var>op=<expr>` means `<var> = <var> op <expr>`

where `op` is `+`, `-`, `*`, `/`, or `%`

- ▣ Example: `total += 5` is the same as `total = total + 5`

4. Simultaneous (multiple) assignments

- ▣ `<var>, <var>, ..., <var> = <expr>, <expr>, ..., <expr>`

- ▣ `sum, diff = x + y, x - y`

Explore with assignment statements

9

- Examine the ***assignmentsAndLoops.py*** module in your Eclipse project.
- Do the TODO's in it.
 - ▣ I'll demo some of them with you.

Compound assignment and related operators (`+=`, `-=`, `*=`, ...)

□ `a += b` is equivalent to `a = a + b`

```
IDLE 1.2.1
```

```
>>> x = 5
```

```
>>> x += 6; print(x)
```

```
11
```

```
>>> x *= 2; print(x)
```

```
22
```

```
>>> x -= 3; (print x)
```

```
19
```

```
>>> x %= 7; (print x)
```

```
5
```

```
>>> s = "abc"
```

```
>>> s += "d"; print(s)
```

```
abcd
```

```
>>> nums = [1,2,3]
```

```
>>> nums += [4,5]
```

```
>>> print(nums)
```

```
[1,2,3,4,5]
```

Sequence

- A list of things
- For example:
 - [2, 3, 5, 7]
 - ["My", "dog", "has", "fleas"]
- Every **for** loop uses a list.

Definite loops

- Definition

- ▣ **Loop:** a **control structure** for executing a portion of a program **multiple times**

- ▣ **Definite:** Python **knows** how many times to **iterate** the body of the loop

- Syntax:

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

Executes <body> once for every element of <sequence>, with <var> set to that element.

Examples using loops

Loop index

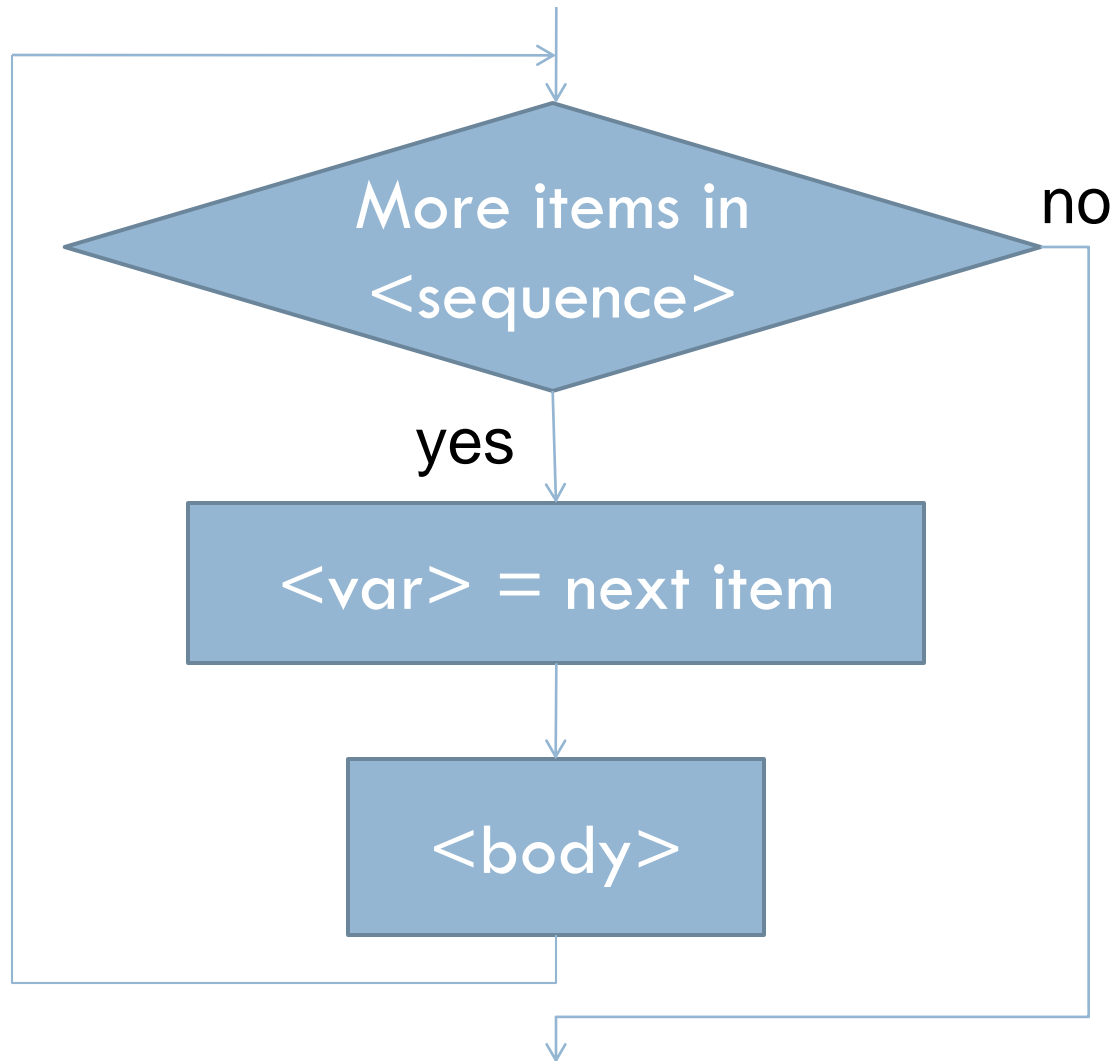
Loop sequence

```
for i in [0, 1, 2, 3, 4, 5]:  
    print(2**i)
```

Loop body

```
for b in ["John", "Paul", "George",  
         "Ringo"]:  
    print(b, " was a Beatle")
```

Flowchart for a for loop



Trace this by hand:

a = 0

for j in [1, 2, 3, 4]:

a = a + j

print(a)

An ***accumulator*** combines parts of a list using looping.

We'll use this idea often this term!

The range function

- A way to create a list that is an arithmetic sequence
- Useful to generate a list used by a for loop
 - ▣ General formats for *range* function:
 - ▣ `range (<expr>)`
 - ▣ `range (<expr>, <expr>)`
 - ▣ `range (<expr>, <expr>, <expr>)`
- What do the following **range** calls do?
- `print(list(range(8)))` `print(list(range(1, 7)))`
`print(list(range(3, 18, 2)))` `print(list(range(4, 10, -1)))`
`print(list(range(17, -5, -3)))`

Use range to make the list for a loop

- `for i in range(7):`
 `print(i, i*i)`
- `for i in range(15, 2, -1):`
 `print(I, end=" ")`
 `print()`

Another loop with an accumulator

- Find the sum of the positive odd numbers that are ≤ 13
- Do it together as a class, in function **sumOddPositiveLessThan()** in

More math library components

Python	Mathematics	English
pi	π	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
atan2(y, x)	$\tan^{-1} y/x$	Arc tangent (inverse tangent) of angle of line from (0,0) to (x, y)
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

Math library functions

Quadratic formula to find real roots for quadratic equations of the form $ax^2 + bx + c = 0$

□ Solution:

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

- Write out the Python expression for the first formula.
- If time permits, test it in Eclipse

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, this technique
 - ▣ Reduces number of errors
 - ▣ Saves time in the long run
- Works best when partners have similar skill level
- If not, then student with most experience should navigate, while the other student drives.

Food tasting

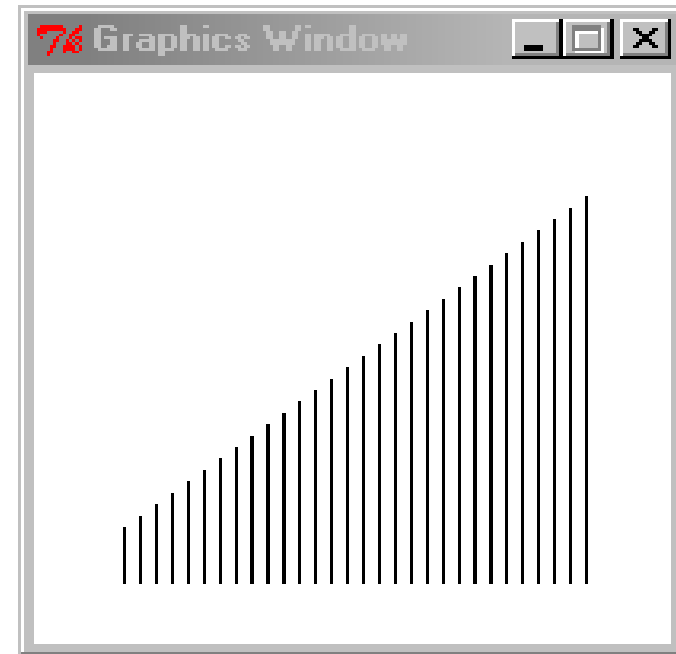
- Suppose you are at food tasting show and are tasting 5 different dishes
- Sampling the dishes in different orders may affect how good they taste
- If you want to try out every possible ordering, how many different orders would there be?
 - ▣ That number is the factorial of 5
 - ▣ $n! = n (n - 1) (n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

Accumulating results: factorial

- Work with a partner (pick a driver and navigator)
- Write a Python program in **factorial.py** that
 - Prompts the user for an integer
 - Calculates the factorial of the integer
 - $n! = n (n - 1) (n - 2) \dots (1)$
 - Does **not** use the built-in `math.factorial` function
 - Outputs the result to the screen
- Driver: email the code to your partner (so each has the program for the open-computer parts of exams)
- Submit one copy of program with both student's names in a program comment.
- Commit your solution to you SVN repository

Graphics Exercise with loops

- Trade roles with partner—new driver, new navigator
- Write a program in **barChart.py** that draws a figure like this where the lengths of the lines increase by a constant amount
- Use your previous graphics program to model 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
- Commit your solution to SVN.



If you don't finish

Factorial or Bar Chart program

- Meet before next class to finish them
- Reminders:
 - ▣ Driver: email the code to your partner (so each has the program for the open-computer parts of exams)
 - ▣ Submit one copy of program with both student's names in a program comment.
 - ▣ Log into Angel and go to the class's webpage
 - ▣ Click on the Lessons tab then go to **Homework** > **Homework 3**
 - ▣ Commit the factorial program to your SVN repository
 - ▣ Commit the line drawing program to your SVN repository