

Assignments and Loops

Rose-Hulman Institute of Technology

Computer Science and Software Engineering

Outline

- Basic number types: **int** and **float**
- Variables and assignments
- Definite loops
- Math library
- Accumulator problem

Check out 03-AssignmentsAndLoops

- Go to SVN Repository view at bottom of the workbench
 - Missing? Add it back:
Window→**Show View**→**Other**→**SVN** → **SVN Repositories**
- Browse SVN Repository view for **03-AssignmentsAndLoops**
- Right-click it and choose **Checkout**
- Accept defaults in the dialog
- Expand the **03-AssignmentsAndLoops** project that appears in **Package Explorer** (on the left-hand-side)

PyDev Interpreter Console

The image shows a sequence of steps to set up and use the PyDev Interpreter Console in Eclipse. Red boxes with numbers 1 through 5 indicate the steps:

1. The 'Console' tab is selected in the Eclipse IDE.
2. The 'New Console View' menu option is selected.
3. The 'Pydev Console' option is selected from the dropdown menu.
4. The 'Python console' option is selected in the dialog box.
5. The 'OK' button is clicked to confirm the selection.

The resulting Pydev Console window shows the following output:

```
Pydev Console [1]
>>> import sys; print('%s %s' % (sys.executable or sys.platform, sys.version))
/Library/Frameworks/Python.framework/Versions/3.1/Resources/Python
[GCC 4.0.1 (Apple Inc. build 5493)]
>>> print("hello!")
hello!
>>> |
```

A red box with the text "Woot! Interpreter Shell" is positioned below the console output.

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

- Identifiers that refer to values stored in memory
- Values can be of any type

```
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 and Assignment

- Assignment gives a variable a value

$$x = 6 * 7$$

- Python evaluates right-hand side (42)
- Then variable on left “gets” the value

- “Gets” not “Equals”

$$x = 3.9 * x * (1 - x)$$

How to Think About Variables

- Variables as sticky notes
- Example on board...

$$x = 10$$

$$x = x + 1$$

Three Kinds of Assignment

- Simple
- Compound
- Multiple (or simultaneous)

Simple Assignment

- `<variable> = <expr>`
- Note:
 - `input(<string>)` is an expression
 - input statements are just assignment statements

Q1,2

Compound Assignment

- $\langle \text{var} \rangle \langle \text{op}=\rangle \langle \text{expr} \rangle$ means
 $\langle \text{var} \rangle = \langle \text{var} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$
 - where $\langle \text{op}=\rangle$ is $+=$, $-=$, $*=$, $/=$, $//=$, or $\%=$
- Example:
 - **total += 5** is the same as
total = total + 5

Simultaneous Assignment

- $\langle \text{var} \rangle, \langle \text{var} \rangle, \dots = \langle \text{expr} \rangle, \langle \text{expr} \rangle, \dots$
- Example:
 - $\text{sum}, \text{diff} = x + y, x - y$

Assignment Assignment

- See `assignmentsAndLoops.py` module
- Do the TODOs inside the `assignmentStatements()` function

Summary: Assignment Statements

- Simple assignments: $\langle \text{variable} \rangle = \langle \text{expr} \rangle$
- Compound assignments
 - $\langle \text{var} \rangle \langle \text{op} \Rightarrow \rangle \langle \text{expr} \rangle$ means
 $\langle \text{var} \rangle = \langle \text{var} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$
where $\langle \text{op} \Rightarrow \rangle$ is $+=$, $-=$, $*=$, $/=$, $//=$, or $\%=$
- Simultaneous (multiple) assignments
 - $\langle \text{var} \rangle, \langle \text{var} \rangle, \dots = \langle \text{expr} \rangle, \langle \text{expr} \rangle, \dots$

Sequence

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

Definite Loops

- *Loop*: a control structure for executing a portion of a program multiple times
- *Definite loop*: Python knows beforehand how many times to repeat the body of the loop
- Syntax:
 for <var> in <sequence> :
 <body>
- Semantics: Executes **<body>** once for every element of **<sequence>**, with **<var>** set to that element.

Some Definite Loops

The diagram illustrates two Python loops with callouts identifying their components. The first loop is a range-based loop where the variable `i` is the loop index, the list `[0, 1, 2, 3, 4, 5]` is the loop sequence, and `print(2**i)` is the loop body. The second loop is a sequence-based loop where the variable `b` is the loop index, the list `["John", "Paul", "George", "Ringo"]` is the loop sequence, and `print(b, "was a Beatle")` is the loop body.

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

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

The range Function

- Creates a list that is an *arithmetic sequence*
- General formats for range function:
 - range(<expr>)
 - range(<expr>, <expr>)
 - range(<expr>, <expr>, <expr>)

- Consider:

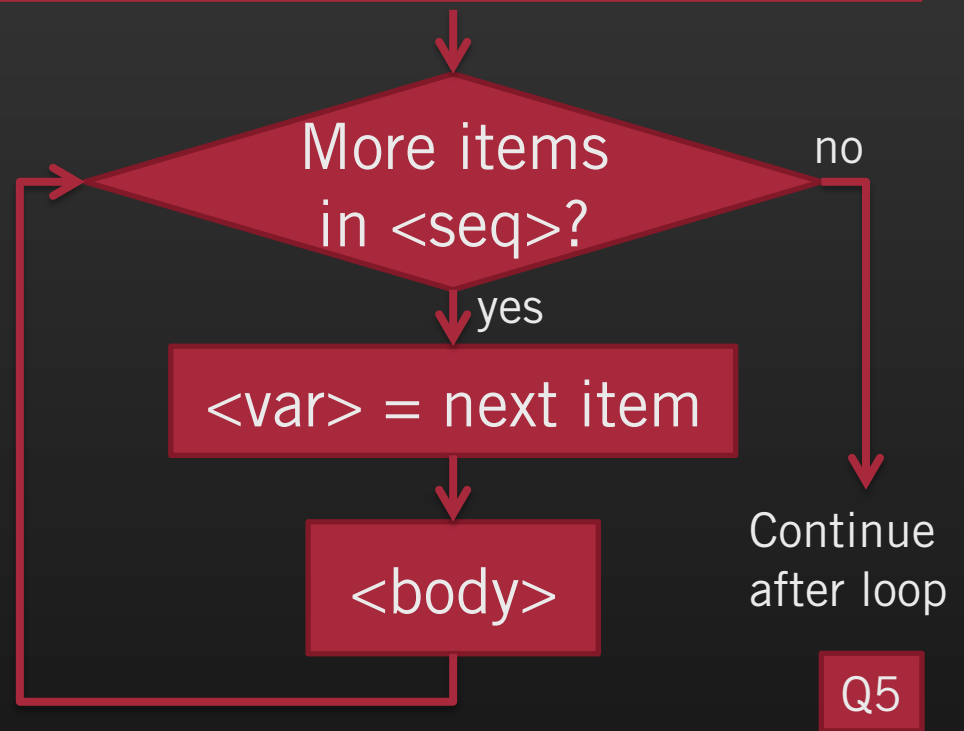
```
list(range(8))  
list(range(1, 7))  
list(range(3, 18, 2))  
list(range(4, 10, -1))  
list(range(17, -5, -3))
```

Q4

Accumulator Loop

- Accumulator combines parts of a list
- Common technique!
- Consider:

```
a = 0
for j in [1, 2, 3, 4]:
    a = a + j
    print(a)
```



Q5

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 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



Reference!

Math library functions

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

- Solution:

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

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

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

Q6

Work Time

HW2 due Mon., HW3 due Tues.