

More Basic Python Programs, Assignments and Loops

Rose-Hulman Institute of Technology

Computer Science and Software Engineering

Outline

- Software development process
- Names, Expressions, output statements
- 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)

The Software Development Process

Analyze the Problem

Determine Specifications

Create a Design

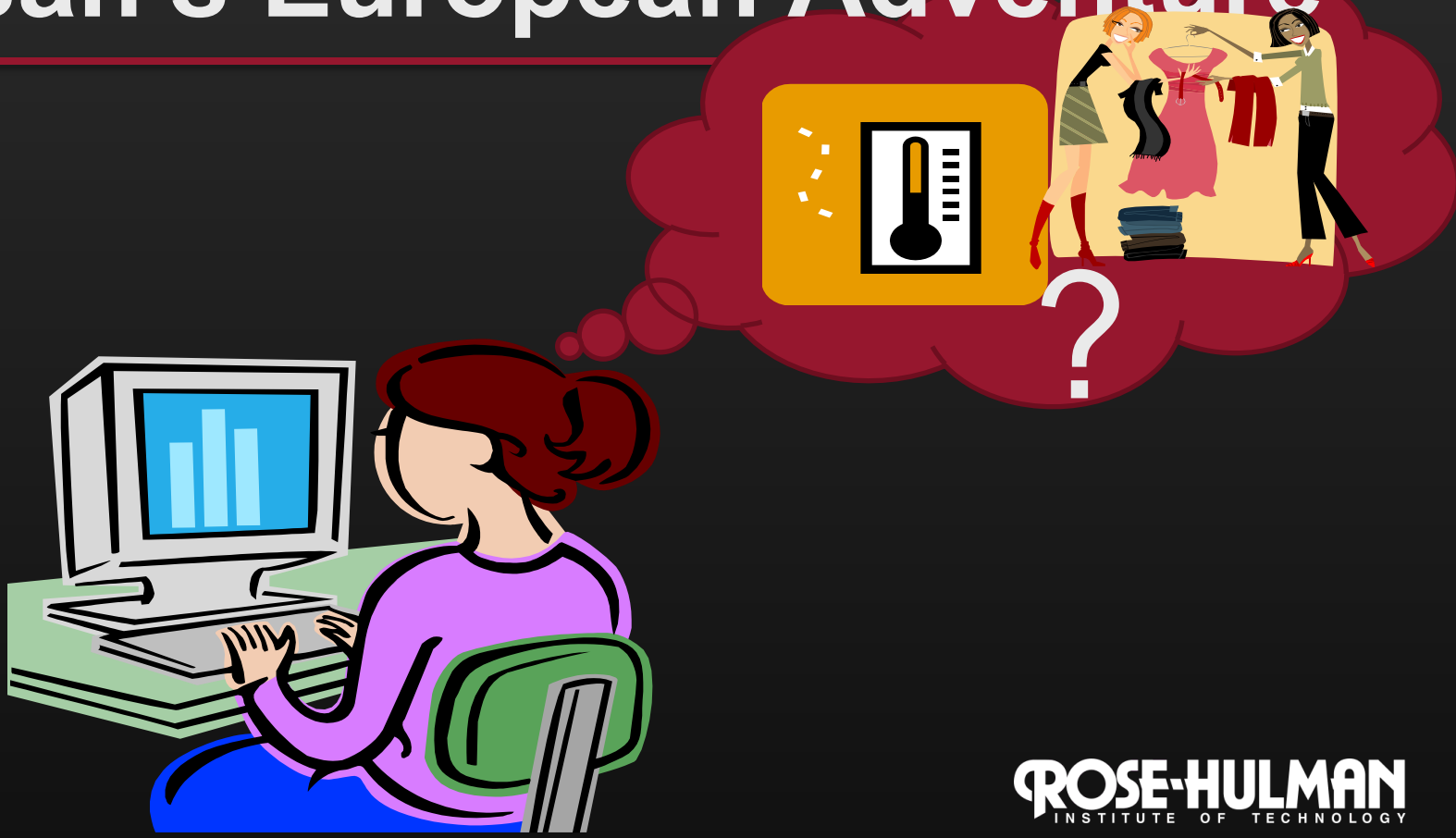
Implement the Design

Test/Debug the Program

Maintain the Program

Q1

Susan's European Adventure



The Software Development Process

Analyze the Problem

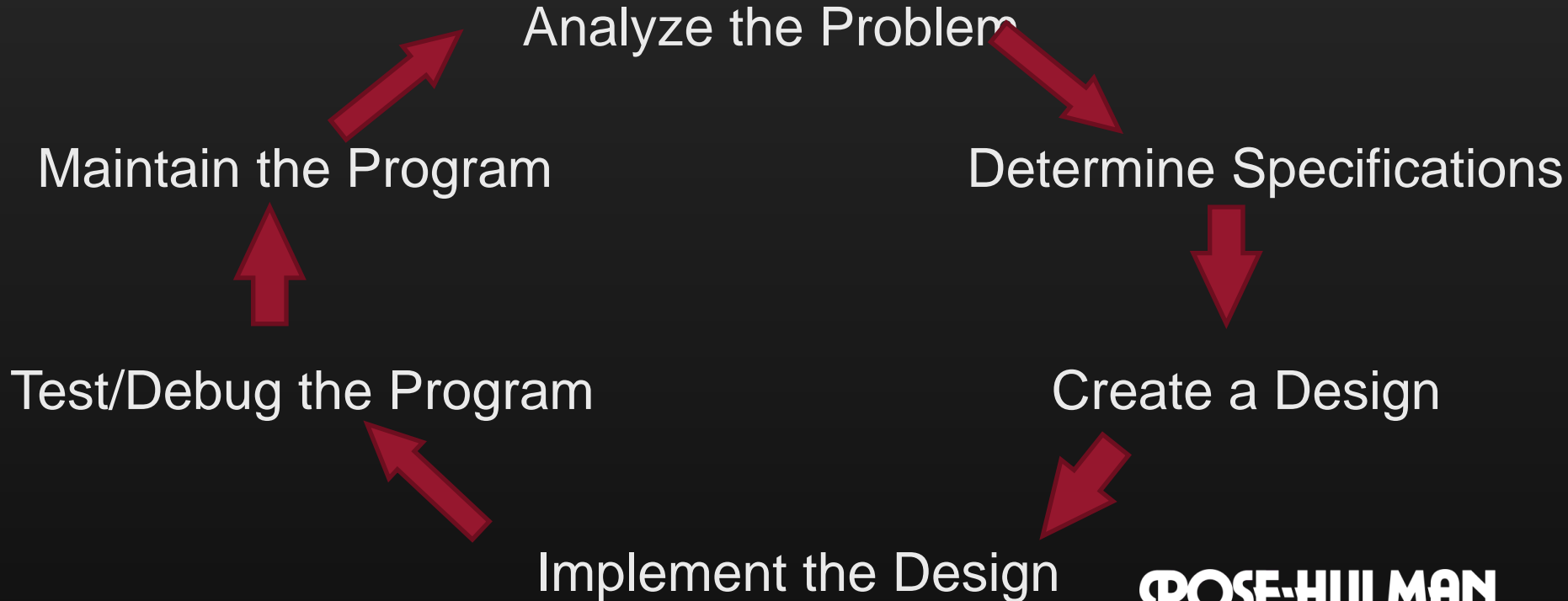
Determine Specifications

Create a Design

Implement the Design

Test/Debug the Program

Maintain the Program



More input-compute-output practice

- Examine the **temperature.py** module in your Eclipse project
- Do the TODOs in it

PyDev Interpreter Console

1

2

3

4

5

Woot!
Interpreter
Shell

1 New Console View
2 Java Stack Trace Console
3 SVN Console
4 CVS
5 Pydev Console

No consoles to display at this time.

Problems SVN Repositories Console Tasks

Unable to create console for current editor (no active editor)
 Python console
Unable to create console for Jython (interpreter not configured)
Unable to create console for Iron Python (interpreter not configured)
Jython using VM running Eclipse console

Cancel OK

Problems SVN Repositories Console Tasks

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

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

Identifiers: Names in Programs

- Uses of identifiers so far...
 - Modules
 - Functions
 - Variables
- Rules for identifiers in Python
 - Start with a letter or `_` (the “underscore character”)
 - Followed by any sequence of letters, numbers, or `_`
- Case matters! **spam** ≠ **Spam** ≠ **sPam** ≠ **SPAM**
- Choose descriptive names!

Q2a

Reserved Words

- Built-in names, can't be used as identifiers
- Python reserved words:

and	del	for	is	raise
assert	elif	from	lambda	return
break	else	global	not	try
class	except	if	or	while
continue	exec	import	pass	with
def	finally	in	print	yield

Q2b

Don't redefine function names!

- Examples that trip me up:
 - *len* – used to find the length of a list
 - *max*
 - *min*

Expressions

- Fragments of code that produce or calculate new data values
- Examples
 - *Literals*: indicate a specific value: **2**, **3.5**, **"hello"**
 - *Identifiers*: evaluate to their assigned value: **x**, **width**
 - *Compound expressions* contain other expressions:
 - E.g., **3 + 4 * 2**, **x < 5**
- Can use parentheses to group: **(3 + 4) * 2**

Q3,4

Recall...

- Programming languages have precise rules for:
 - Syntax (form)
 - Semantics (meaning)
- Computer scientists use *meta-languages* to describe these rules
- Example...

Output Statements

- Syntax:
 - `print()`
 - `print(<expr>)`
 - `print(<expr>, ...)`
 - `print(<expr>, ..., end=<str>)`
- Are these allowed?
 - `print("The answer is:", 7 * 3 * 2)`
 - `print(3, 5, 7, end="")`
- Semantics?

A “slot” to be filled with any expression

Repeat indefinitely as many times as desired, zero or more

Q5

Variables

- Identifiers (i.e.names) 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 a kind of assignment statement

Q6,7

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:
 - $\text{total} += 5$ is the same as
 $\text{total} = \text{total} + 5$

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
 - $\langle \text{var} \rangle \langle \text{op}=\rangle \langle \text{expr} \rangle$ means
 $\langle \text{var} \rangle = \langle \text{var} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$
where $\langle \text{op}=\rangle$ is $+=$, $-=$, $*=$, $/=$, $//=$, or $\%=$
- 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

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")
```

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))
```

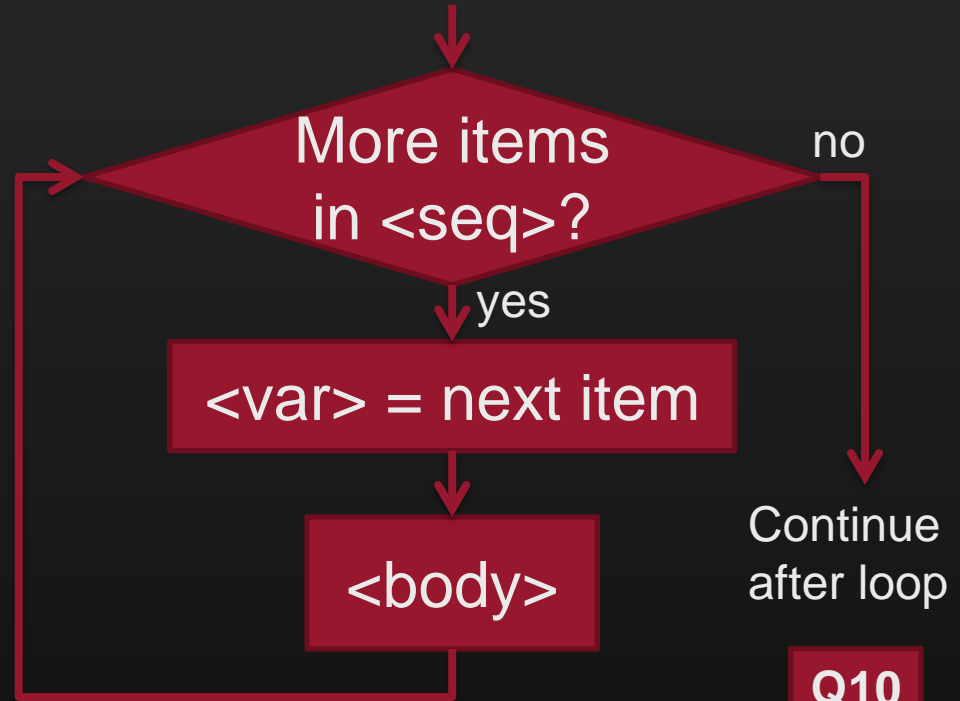
```
list(range(4, 10, -1))
```

```
list(range(17, -5, -3))
```

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)
```



Q10

Another loop with an accumulator

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

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

Q11

Work Time

**HW2 due Monday at 8:55 AM,
HW3 due Tuesday at 8:55 AM**

Q12-13