

Objectives

- Wrap up arithmetic
- A few programming tricks
- Intro to design patterns
- Definite loops

Jan 18, 2012

Sprenkle - CSCI111

1

Lab Review

- Batch mode vs interpreted mode

Jan 18, 2012

Sprenkle - CSCI111

2

Modulo Operator: %

- Modular Arithmetic: Remainder from division
 - $x \% y$ means the remainder of x/y
 - Read as "x mod y"
- Example: $6 \% 4$
 - Read as "six mod four"
 - $6/4$ is 1 with a remainder of 2, so $6\%4$ evaluates to 2
- Works only with integers
 - Typically just positive numbers
- Precedence rules: P E - DM% AS

Jan 18, 2012

Sprenkle - CSCI111

3

Modulo Practice

- $7 \% 2$
- $3 \% 6$
- $6 \% 2$
- $7 \% 14$
- $14 \% 7$
- $6 \% 0$

Jan 18, 2012

Sprenkle - CSCI111

4

Brainstorm

- What useful thing does $\% 10$ do?
 - $3 \% 10 =$
 - $51 \% 10 =$
 - $40 \% 10 =$
 - $678 \% 10 =$
 - $12543 \% 10 =$
- What useful thing does $// 10$ do (integer division)?
 - $3 // 10 =$
 - $51 // 10 =$
 - $40 // 10 =$
 - $678 // 10 =$
 - $12543 // 10 =$
- What useful thing does $\% 2$ do?

Jan 18, 2012

Sprenkle - CSCI111

5

Trick #1: Type Conversion

- You can convert a variable's type
 - Use the type's **constructor**

Conversion Function/ Constructor	Example	Value Returned
<code>int(<number or string>)</code>	<code>int(3.77)</code> <code>int("33")</code>	3 33
<code>float(<number or string>)</code>	<code>float(22)</code>	22.0
<code>str(<any value>)</code>	<code>str(99)</code>	"99"

Jan 18, 2012

Sprenkle - CSCI111

6

Example Using Type Conversion

- May want to restrict the type of values that a user enters
- For example, a user's age should be an integer

```
orig_age = eval(input("What is your age? "))
int_age = int(orig_age)
print("Your age is", int_age)
```

Ideally, we'd tell the user that we made a change to their input, but we don't know how to do that yet.

Jan 18, 2012

Sprenkle - CSC1111

7

Another Example: Restricting User's Inputs

```
>>> x = 7
>>> yourVal = input("My val is: ")
My val is: x
>>> print(yourVal)
x
>>> yourVal = eval(input("My val is: "))
My val is: x
>>> print(yourVal)
7
>>> yourVal = int(input("My val is: "))
My val is: x
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: 'x'
```

Jan 18, 2012

Sprenkle - CSC1111

8

Trick #2: Arithmetic Shorthands

- Called **extended assignment operators**
- Increment Operator
 - $x = x + 1$ can be written as $x += 1$
- Decrement Operator
 - $x = x - 1$ can be written as $x -= 1$
- Shorthands are similar for $*$, $/$, $//$:
 - $\text{amount} *= 1.05$
 - $x //= 2$

Jan 18, 2012

Sprenkle - CSC1111

9

FOR LOOPS

Jan 18, 2012

Sprenkle - CSC1111

10

Parts of an Algorithm

- Input, Output
- Primitive operations
 - What data you have, what you can do to the data
- Naming
 - Identify things we're using
- Sequence of operations
- Conditionals
 - Handle special cases
- Repetition/Loops 
- Subroutines
 - Call, reuse similar techniques

Jan 18, 2012

Sprenkle - CSC1111

11

Looping/Repetition

We know how to make a PB&J Sandwich: Make PB&J sandwich

Make 10 PB&J sandwiches {

- Make PB&J sandwich

Repetition is common in programming. Is there some simpler way to say that we want to repeat something?

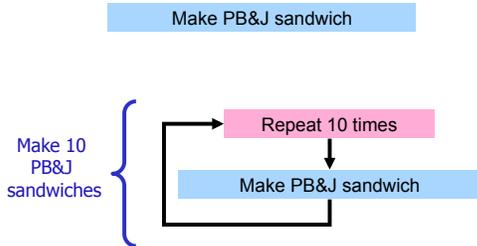
Make PB&J sandwich

Jan 18, 2012

Sprenkle - CSC1111

12

Looping/Repetition



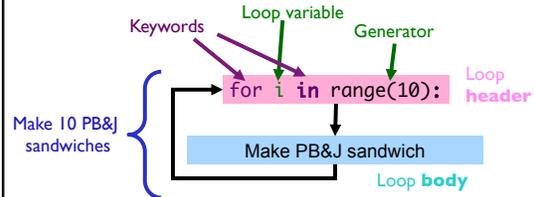
Jan 18, 2012

Sprengle - CSCI111

13

The for Loop

- Use when know how many times loop will execute
 - Repeat N times



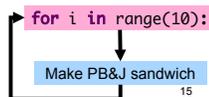
Jan 18, 2012

Sprengle - CSCI111

14

What Goes in the Loop Body?

- Make PB&J Sandwich
 - Gather materials (bread, PB, J, knives, plate)
 - Open bread
 - Put 2 pieces of bread on plate
 - Spread PB on one side of one slice
 - Spread Jelly on one side of other slice
 - Place PB-side facedown on Jelly-side of bread
 - Close bread
 - Clean knife
 - Put away materials



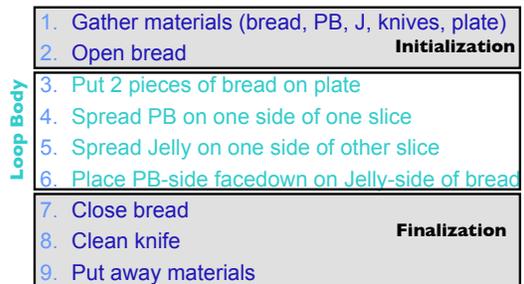
Jan 18, 2012

Sprengle - CSCI111

15

What Goes in the Loop Body?

- Make PB&J Sandwich



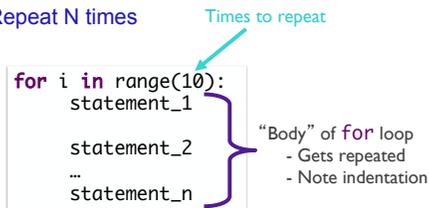
Jan 18, 2012

Sprengle - CSCI111

16

for Loop Syntax and Semantics

- Use when know how many times loop will execute
 - Repeat N times



Jan 18, 2012

Sprengle - CSCI111

17

Using the for Loop

- If only **one** statement to repeat

```
for variable in range(5): print("Hello!")
```

Jan 18, 2012

Sprengle - CSCI111

simple_for.py 18

Analyzing range()

- **range** is a *generator*
- What does **range** do, exactly, with respect to the loop variable *i*?

```
for i in range(10):  
    squared = i * i  
    print(i, "^2 =\t", squared)  
  
print(i)
```

range_analysis.py

Jan 18, 2012

Sprengle - CSC1111

19

range([start,] stop[, step])

- What does the above signature mean?

Jan 18, 2012

Sprengle - CSC1111

20

range([start,] stop[, step])

- 1 argument: range(stop)
- 2 arguments: range(start, stop)
- 3 arguments: range(start, stop, step)

Jan 18, 2012

Sprengle - CSC1111

using_range.py

21

range([start,] stop[, step])

- 1 argument: range(stop)
 - Defaults: start = 0, step = 1
 - Iterates from 0 to stop-1 with step size=1
- 2 arguments: range(start, stop)
 - Default: step = 1
 - Iterates from start to stop-1 with step size=1
- 3 arguments: range(start, stop, step)
 - Iterates from start to stop-1 with step size=step

Jan 18, 2012

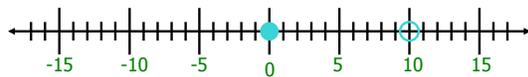
Sprengle - CSC1111

using_range.py

22

range

- **range** is a number generator
 - 1 argument: range(stop)
 - 2 arguments: range(start, stop)
 - 3 arguments: range(start, stop, step)



```
xrange(10)  
xrange(0,10)  
xrange(0,10,1)
```

[start, stop)

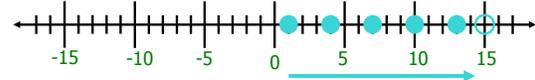
Jan 18, 2012

Sprengle - CSC1111

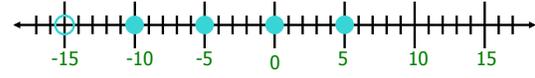
23

Sequence generated by range

range(1, 15, 3):



range(5, -15, -5):



more_range_examples.py

Jan 18, 2012

Sprengle - CSC1111

24

Practice

Place these: ● ○
Which direction?

`range(2, 14, 2):`

`range(8, -10, -3):`

`range(-5, 15, -3):`

Jan 18, 2012 Sprenkle - CSCI111 25

Practice Solution

`range(2, 14, 2):`

`range(8, -10, -3):`

`range(-5, 15, -3):`

Jan 18, 2012 Sprenkle - CSCI111 26

Practicing for Loops

- Write the Python code to print the following:
 - > A)


```
1
2
3
4
5
```
 - > B)


```
2
5
8
11
```
 - > C)


```
****
****
****
```

What is getting repeated?
How many times?

Jan 18, 2012 Sprenkle - CSCI111 27

Programming Practice

- Add 5 numbers, inputted by the user
 - > After implementing, simulate running on computer

Jan 18, 2012 Sprenkle - CSCI111 `sum5.py` 28

Generalizing Solution: Accumulator Design Pattern

1. Initialize accumulator variable
2. Loop until done
 - > Update the value of the accumulator
3. Display result

Jan 18, 2012 Sprenkle - CSCI111 29

Programming Practice

- Average 5 numbers inputted by the user

Jan 18, 2012 Sprenkle - CSCI111 `average5.py` 30

Designing for Change

- What are we likely to change in the program?
- How can we make the program easier to change?

Jan 18, 2012

Sprenkle - CSCI111

31

For This Week

- Lab 1: Due Friday by classtime
- Broader Issues: Four Puzzles from Cyberspace
 - Through "Themes"
 - Posted on Sakai by 10 a.m. on Friday

Jan 18, 2012

Sprenkle - CSCI111

32