Loops and
Simple Functions

CS303E: Elements of Computers
and Programming
June 20, 2012

Review:
While Loops

- Typically used when the number of times the loop will execute is *indefinite*
- General pseudocode:

```
read the first value
while value is not the sentinel:
    process the value
    read the next value
```

More Loops

- *Definite* loops
  - Execute the body of the loop a specific number of times
  - Use a counter
  - `for` loop
- As opposed to *indefinite* loops like the `while` loop

For Loops:
Syntax

Loop Index

Indentation Matters!

```
for <variable> in <sequence>:
    statement
    statement
    …
```

Loop Body

The *Loop Index* takes on the values in the sequence, one at a time, and the *Loop Body* is executed once for each value

For Loops: Example

```
for i in [1,2,3,4]:
    print i
```

Loop Index

Programmers often use *i* or *j*

Output:

1
2
3
4

Example of Sequence format:

Begin and end with square brackets
Separate values with a comma

Called a *list*

For Loops: Example

```
for num in [3,2,1,5]:
    print num,
```

iClicker Question

What is the output?

```
for j in [2,4,6,8]
    print j,
```

A. 1 2 3 4   C. 1,2,3,4
B. 2 4 6 8   D. 2,4,6,8

For Loops: What about range()?

`range(<expr>)` is a *function*
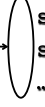- Produces a list of numbers
- Several variations of <expr> accepted

range(<expr>)	Result
range(n)	A list of integers from 0 to n-1
range(n,m)	A list containing the integers from n to m-1
range(n,m,step)	A list containing n, n+1*step, n+2*step, n+3*step,… up to m-1

More range()

- range() stops at:
  - One *less* than `end` if the step is positive
  - One *more* than `end` if the step is negative
- Generalization, check documentation for formal definition

For Loops: Example

```
for i in range(2,6):
    print i, i**2
```

For Loops: Exercise

Write a for loop that prints the even numbers from 2 to 20 (inclusive). The output should appear on one line, with the numbers separated by a blank space

Functions: What are they?

- Statements grouped under a special name that are executed together
- Useful to execute statements from different parts of the program
- Advantages
  - Code reuse
    - Type once, use again and again!
    - Easier to maintain (update, fix mistakes)
  - Code readability

Functions: Syntax

Indentation Matters!

```
def functionName():
    statement
    statement
    …
```

Function Body

Functions: Syntax

Recall:
```
def functionName():
```

To *call* that function, type its name:
```
functionName()
```
- When a function is called the statements in the body are executed
- If the function is not called, the statements are never executed

Functions: Examples

- We've seen:
  - `main():` We've *defined* `main()`
  - `range(<expr>):` We've *called* `range();` it is built-in
  - `random.randint(<expr>):` We've *called* `randint();` it is located in the module random
- Queen.py
- Queen_readable.py

iClicker Question: Functions

Why are functions useful?

A. Improve code readability
B. Reduce code maintenance
C. Both A and B
D. Neither A or B

Reminders

- Assignment 2 due tomorrow
- Exam is NEXT Friday