Objectives

- Code Readability
- Intro to conditional statements
- **sys** module
- Broader Issue: 4 Puzzles from Cyberspace

What Does This Program Do?

```
import random

winningNum = ""

for x in xrange(3):
    numChosen = random.randint(0,9)
    winningNum += str(numChosen) +"-"

numChosen = random.randint(0,9)
winningNum += str(numChosen)

print "The number is", winningNum
```


VA Lottery: Pick 4

- To play: you pick 4 numbers between 0 and 9
- To win: select the numbers that are selected by the magic ping-pong ball machine
- Your job: Simulate the magic ping-pong ball machines
  - Revision: display number as #-#-#-#

pick4.nocomments.py

VA Lottery: Mega Millions

- Modify your Pick 4 to simulate Mega Millions
- To play: you pick 5 numbers between 1 and 56
  - Ignoring rule: 1 Mega Ball number between 1 and 46
- Your job: Simulate the result of the magic ping-pong ball machines, displayed as #-#-#-#-#
  - How difficult to modify the last program?
  - What could we do to make easier?

Changes to pick4.py

- Comments
  - Clarify what the program is doing
  - We wrote the program Wednesday
    - Already unclear on the details
- Constants
  - Give ***meaning*** to "magic numbers"
    - What were 0, 9, 3?

Improving Code Readability

- Comments
  - Describe blocks of code at a high level
- Output/Display
  - Descriptive, explains what program outputs
- Constants
  - Change one value (at top of program) to change value everywhere in program
  - Flexible programs
  - Gets rid of "magic numbers"
    - Give a clear name and purpose to values

Improving Code Readability/Usability

- What does this program do?
  - How would you figure it out?
- What would you do to improve the program's readability and usability?

program_before.py
program_after.py

Comparing Programs

- `constant_compare.out`
- Note good use of comments
  - Define sections of code
- Compare with and without constants

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

Making Decisions

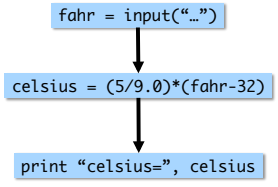
- Sometimes, we do things only if some other condition holds (i.e., "is true")
- Examples
  - If the PB is new (has a safety seal)
    - Then, I will take off the safety seal
  - If it is raining and it is cold
    - Then, I will wear a raincoat
  - If it is Saturday or it is Sunday
    - Then, I will wake up at 10 a.m.
    - Otherwise, I wake up at 7 a.m.
  - If the shirt is purple or the shirt is on sale and blue
    - Then, I will buy the shirt

Conditionals

- Sometimes, we only want to execute a statement in certain cases
  - Example: Finding the absolute value of a number
    - |4| = 4
    - |-10| = 10
  - To get the answer, we multiply the number by -1 *only if it's a negative number*
  - Code:

```
if x < 0 :
    abs =  x*-1
```


Typical Execution

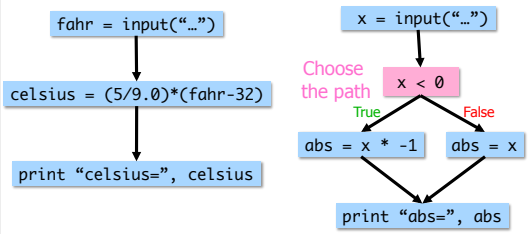


So far, we've thought of programs as a *sequence* of statements.

Statements execute *in order*.

if Statements

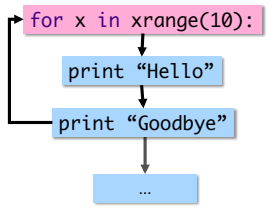
- Change the ***control flow*** of the program



Other "things" that change control flow

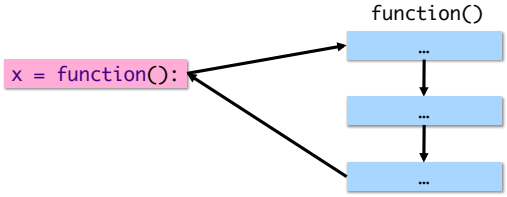
- **for** loops
  - Repeats a loop body a fixed number of times before going to the next statement after the **for** loop

```
for x in xrange(10):
    print "Hello"
    print "Goodbye"
next_statement …
```



Other "things" that change control flow

- Function calls
  - "Go execute some other code and then come back with the result"



Syntax of if statement: Simple Decision

```
if condition :
    statement1
    statement2
    …
    statementn
```

keyword

"then" Body
- Note indentation

English Examples:

if it is raining **:**
    I will wear a raincoat
if the PB is new **:**
    Remove the seal

Conditions

- Syntax:
  - `<expr> <relational_operator> <expr>`
- Evaluates to either `True` or `False`
  - Boolean type

Relational Operators

- Syntax:
  - `<expr> <relational_operator> <expr>`

Relational Operator	Meaning
<	Less than?
<=	Less than or equal to?
>	Greater than?
>=	Greater than or equal to?
==	Equals?
!=	Not equals?

Examples: Using Conditionals

- Determine if a number is even or odd

```
x = input("Enter a number: ")
remainder = x%2
if remainder == 0 :
    print x, "is even"
if remainder == 1:
    print x, "is odd"
```


Common Mistake: Assignment Operator vs. Equality Operator

- Assignment operator: =
- Equality operator: ==

```
x = input("Enter a number: ")
remainder = x%2
if remainder = 0 :
    print x, "is even."
```

Syntax error

Syntax of `if` statement: Two-Way Decision

```
if condition :
    statement1
    statement2
    …
    statementn
else :
    statement1
    statement2
    …
    statementn
```

keywords

"then" Body

"else" Body

English Example:

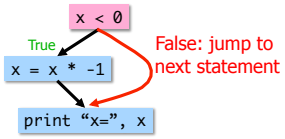
```
if it is Saturday or Sunday :
    I wake up at 10 a.m.
else :
    I wake up at 7 a.m.
```


If-Else statements (absolute values)

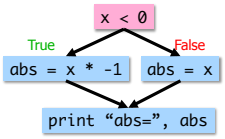
```
if x < 0 :
    x *= -1
print "x=", x
```

```
if x < 0 :
    abs = x * -1
else :
    abs = x
print "abs=", abs
```

If statement



If-else statement



Examples: Using Conditionals

- Determine if a number is even or odd
- More efficient implementation
  - Don't need to check if remainder is 1 because if it's not 0, it must be 1

```
x = input("Enter a number: ")
remainder = x % 2
if remainder == 0:
    print x, "is even"
else:
    print x, "is odd"
```


Practice: Draw the Flow Chart

```
print "This program determines your birth year"
print "given your age and current year"
print
age = input("Enter your age >> ")

if age > 110:
    print "Don't be ridiculous, you can't be that old."
else:
    currentYear = input("Enter the current year >> ")
    birthyear = currentYear - age
    print
    print "You were either born in", birthyear, "or",
    print birthyear-1
```

SYS MODULE

sys module

- Has useful "system" functions
- Use the `exit([status])` function
  - **Exits the whole program**
  - If status is empty, defaults to 0
  - Status of 0 means success
  - Other values are various failures
- *Another example of changing control flow*

Example Use of sys module

```
import sys
print "This program determines your birth year"
print "given your age and current year"
print
age = input("Enter your age >> ")
if age > 110:
  print "Don't be ridiculous, you can't be that old."
  sys.exit(1)
# input is reasonable …
currentYear = input("Enter the current year >> ")
birthyear = currentYear - age
print
print "You were either born in", birthyear, "or",
print birthyear-1
```

Ejector seat

Practice: Speeding Ticket Fines

- Any speed clocked over the limit results in a fine of at least $50, plus $5 for each mph over the limit, plus a penalty of $200 for any speed over 90mph.
- Our program
  - Input: speed limit and the clocked speed
  - Output: either (a) that the clocked speed was under the limit or (b) the appropriate fine

Reminders: Exam Next Friday

- Give you "Exam prep document" with the topics/ concepts by Monday
- Format:
  - Very short answer
  - Short answer
  - What does the code do? (output)
  - Writing code
  - Problem-solving
- To Review:
  - In-class problems/handouts
  - Labs

Four Puzzles in Cyberspace

- Context: Book Code v2 by Lawrence Lessig
- You read Chapter 2
  - Presents the problems, not the author's proposed solutions

Collier Dave Harrison CJ	Jeni Nick Shannon James	George Will Hank Kelly Mae	Luke Amy Dalena Logan	Andrew Taylor Ben Phil

Discussion Questions

- What are the four puzzles of Cyberspace?
- What is the most important puzzle to solve?
- Which is the most difficult puzzle to solve?
- Which is the most unsettling puzzle?
- What are other examples of online regulation?
  - How successful were the attempts at regulation?