Unless otherwise noted, the content of this course material is licensed under a Creative Commons Attribution 3.0 License. http://creativecommons.org/licenses/by/3.0/.

Copyright © 2009, Charles Severance.

You assume all responsibility for use and potential liability associated with any use of the material. Material contains copyrighted content, used in accordance with U.S. law. Copyright holders of content included in this material should contact open.michigan@umich.edu with any questions, corrections, or clarifications regarding the use of content. The Regents of the University of Michigan do not license the use of third party content posted to this site unless such a license is specifically granted in connection with particular content. Users of content are responsible for their compliance with applicable law. Mention of specific products in this material solely represents the opinion of the speaker and does not represent an endorsement by the University of Michigan. For more information about how to cite these materials visit http://michigan.educommons.net/about/terms-of-use.

Any medical information in this material is intended to inform and educate and is not a tool for self-diagnosis or a replacement for medical evaluation, advice, diagnosis or treatment by a healthcare professional. You should speak to your physician or make an appointment to be seen if you have questions or concerns about this information or your medical condition. Viewer discretion is advised: Material may contain medical images that may be disturbing to some viewers.
x = 5
print "Before 5"
if ( x == 5 ) :
    print "Is 5"
    print "Is Still 5"
    print "Third 5"
print "Afterwards 5"

print "Before 6"
if ( x == 6 ) :
    print "Is 6"
    print "Is Still 6"
    print "Third 6"
print "Afterwards 6"
Several ifs

faren = 120
if ( faren > 90) :
    print "Heat Warning"

if ( faren < 32 ) :
    print "Cold Warning"
Several ifs

faren = 120
if ( faren > 90) :
    print "Heat Warning"

if ( faren < 32) :
    print "Cold Warning"
Comparison Operators

- Boolean expressions using comparison operators evaluate to - True / False - Yes / No

- Boolean expressions ask a question and produce a Yes or No result which we use to control program flow

- Comparison operators look at variables but do not change the variables

<table>
<thead>
<tr>
<th>Python</th>
<th>Mathematics</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>!=</td>
<td>≠</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>

<expr> <relop> <expr>

http://en.wikipedia.org/wiki/George_Boole
x = 5
if ( x == 5 ) :
    print "Equals 5"

if ( x > 4 ) :
    print "Greater than 4"
if ( x >= 5 ) :
    print "Greater than or Equal 5"

if ( x < 6 ) :print "Less than 6"

if ( x <= 5 ) :
    print "Less than or Equal 5"
if ( x != 6 ) :
    print "Not equal 6"
Review Indentation

• Must increase indent after an if statement or for statement (after : )

• Maintain indent to indicate the scope of the block (which lines are affected by the if/for)

• Reduce indent to back to the level of the if statement or for statement to indicate the end of the block

• Blank lines are ignored - they can appear anywhere

• Comments on a line by themselves are ignored
x = 5
if x > 2:
    print "Bigger than 2"
    print "Still bigger"
    print "Done with 2"
for i in range(5):
    print i
    if i > 2:
        print "Bigger than 2"
        print "Done with i", i
x = 5
if x > 2 :
    print "Bigger than 2"
    print "Still bigger"
print "Done with 2"

for i in range(5) :
    print i
    if i > 2 :
        print "Bigger than 2"
    print "Done with i", i

x = 5
if x > 2 :
# comments
    print "Bigger than 2"
    # don’t matter
    print "Still bigger"
# but can confuse you
print "Done with 2"
# if you don’t line
# them up
Nested Decisions

fline = "blah blah"

if len(fline) > 1 :
    print "More than one"
    if fline[0] == 'b' :
        print "Starts with a b"
print "All done"

print “All Done”
Nested Decisions

fline = "blah blah"

if len(fline) > 1:
    print "More than one"
    if fline[0] == 'b':
        print "Starts with a b"
print "All done"

print "All Done"
Nested Decisions

cline = "blah blah"

if len(cline) > 1 :
    print "More than one"
    if cline[0] == 'b' :
        print "Starts with a b"

print "All done"
Two Way Decisions

- Sometimes we want to do one thing if a logical expression is true and something else if the expression is false.

- It is like a fork in the road - we must choose one or the other path but not both.
Two-way the hard way

\[ x = 4 \]
if \( x > 2 \):
print “Bigger”

if \( x \leq 2 \):
print “Smaller”
Two-way using else:

```python
x = 4
if x > 2:
    print "Bigger"
else:
    print "Not bigger"
print "All done"
```
Two-way using else:

```python
x = 4
if x > 2:
    print "Bigger"
else:
    print "Smaller"
print "All done"
```
if $x < 2$:
    print “Small”
elif $x < 10$:
    print “Medium”
else:
    print “LARGE”
print “All done”
x = 0
if x < 2 :
    print “Small”
elif x < 10 :
    print “Medium”
else :
    print “LARGE”
print “All done”
Multi-way

\[ x = 5 \]

if \( x < 2 \):
    print “Small”
elif \( x < 10 \):
    print “Medium”
else:
    print “LARGE”
print “All done”
Multi-way

\[
x = 20
\]

if \( x < 2 \):
    print “Small”
elif \( x < 10 \):
    print “Medium”
else:
    print “LARGE”
print “All done”
# Multi-way

No Else

```python
x = 5
if x < 2 :
    print "Small"
elif x < 10 :
    print "Medium"
print "All done"
```

```python
if x < 2 :
    print "Small"
elif x < 10 :
    print "Medium"
elif x < 20 :
    print "Big"
elif x < 40 :
    print "Large"
elif x < 100:
    print "Huge"
else :
    print "Ginormous"
```
Which will never print?

```python
if x < 2 :
    print "Below 2"
elif x >= 2 :
    print "Two or more"
else :
    print "Something else"
```
The **try** / **except** Structure

- You surround a dangerous section of code with **try** and **except**.
- If the code in the **try** works - the **except** is skipped.
- If the code in the **try** fails - it jumps to the **except** section.
```
$ cat notry.py
astr = "Hello Bob"
istr = int(astr)
```

```
$ python notry.py
Traceback (most recent call last):
  File "notry.py", line 6, in <module>
    istr = int(astr)
ValueError: invalid literal for int() with base 10: 'Hello Bob'
```

The program stops here.
```python
$ cat tryexcept.py
astr = "Hello Bob"
try:
    istr = int(astr)
except:
    istr = -1
print "First", istr

astr = "123"
try:
    istr = int(astr)
except:
    istr = -1
print "Second", istr

$ python tryexcept.py
First -1
Second 123
```

When the first conversion fails - it just drops into the except clause and the program continues.

When the second conversion succeeds - it just skips the except clause and the program continues.
try / except

astr = "Bob"
try:
    print "Hello"
    istr = int(astr)
    print "There"
except:
    istr = -1
print "Done", istr

print "Hello"
print "Hello"
print "There"
print "There"
istr = int(astr)
istr = int(astr)
print "Done", istr
istr = -1
print "Done", istr

Safety net
Sample try/except

fname = raw_input("Enter a file name: ")
infile = open(fname, "r")
print “Blah...”

$ python frompart.py
Enter a file name: fred
Traceback (most recent call last):
  File "frompart.py", line 7, in <module>
    infile = open(fname, "r")
IOError: [Errno 2] No such file or directory: 'fred'
Sample try/except

define the variable <br>
fname = raw_input("Enter a file name: ")
write code for the try block <br>
try:
  open the file <br>  infile = open(fname, "r")
declare the exception <br>
except:
  if the exception occurs <br>    print "File not found", fname <br>    close the file <br>    exit() <br>print "Blah..."
run the code <br>$ python frompart.py
$ Enter a file name: fred
$ File not found fred
$
Another try/except

def main():
    fname = raw_input("Enter a number: ")
    try:
        ival = int(fname)
    except:
        ival = -1
    if ival > 0:
        print "Nice Work"
    else:
        print "Not a number"

if __name__ == '__main__':
    main()
Summary

- Indentation
- One Way Decisions
- Comparison operators $== <= >= > < !=$
- Nested Decisions
- Two way Decisions if : and else :
- Multiway decisions using elif
- Try / Except to compensate for errors