Lesson3: Control Flow

Fundamentals of Text Processing for Linguists

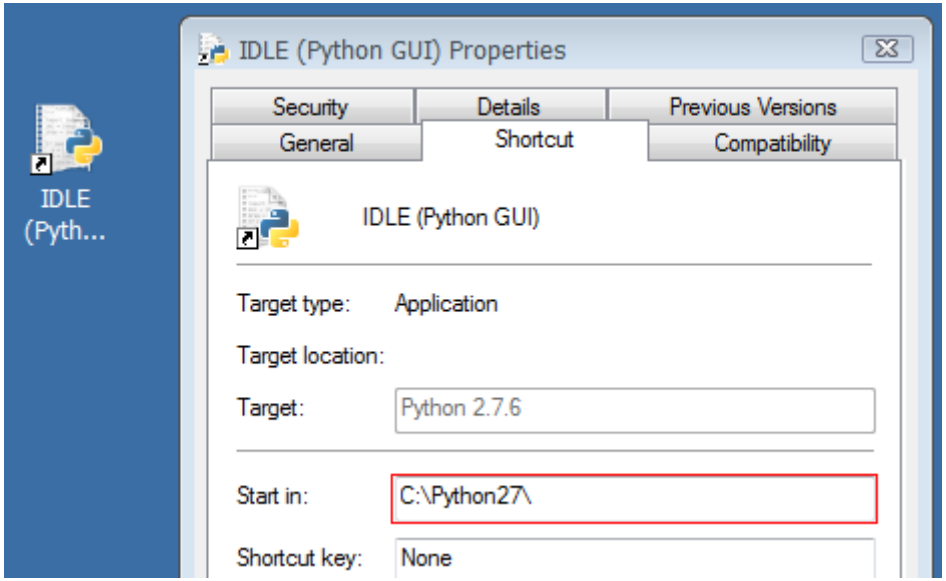
Na-Rae Han

Objectives

- Learn Python basic syntax
  - Taking input from keyboard
  - Commenting
  - Comparison operators
- Control flow
  - Conditional statement: If… elif… else
  - Python indentation
- Splitting and joining
  - Splitting strings and joining lists

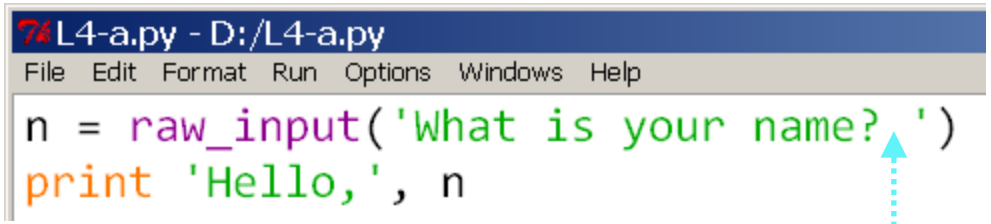
Changing script file location

- IDLE saves and looks for a script file in a default directory.
  - Windows 7: It is **C:\Python27**.
    - NOT a convenient place for your scripts!
    - Change this on a shortcut icon's property:
  - OS X: ?



Taking input from keyboard

- `raw_input('prompt␣')`
  - Prompts for user keyboard input, stores value
- IDLE editor window (save as "hello-name.py"):



- IDLE shell:

```
>>>
What is your name? Charlie Brown
Hello, Charlie Brown
>>>
```

Space '␣' at the end of the string separates the prompt from keyboard input

Try it out

1 minute

- `raw_input('prompt␣')`
- Another script (name it "noun-verb.py")

L4-b.py - D:/L4-b.py

File Edit Format Run Options Windows Help

```
noun = raw_input('What is your noun? ')
verb = raw_input('What is your verb? ')
sent = noun + ' ' + verb
print 'Your sentence is:'
print sent
```

- Execution:

```
>>>
What is your noun? cats
What is your verb? sleep
Your sentence is:
cats sleep
>>>
```

Commenting

- Comments start with "#" ('pound' or 'hash')
  - They are ignored by Python interpreter
  - Works on a single line only
  - Good for notes, explanation, or other information for human readers

L4-b.py - D:/L4-b.py

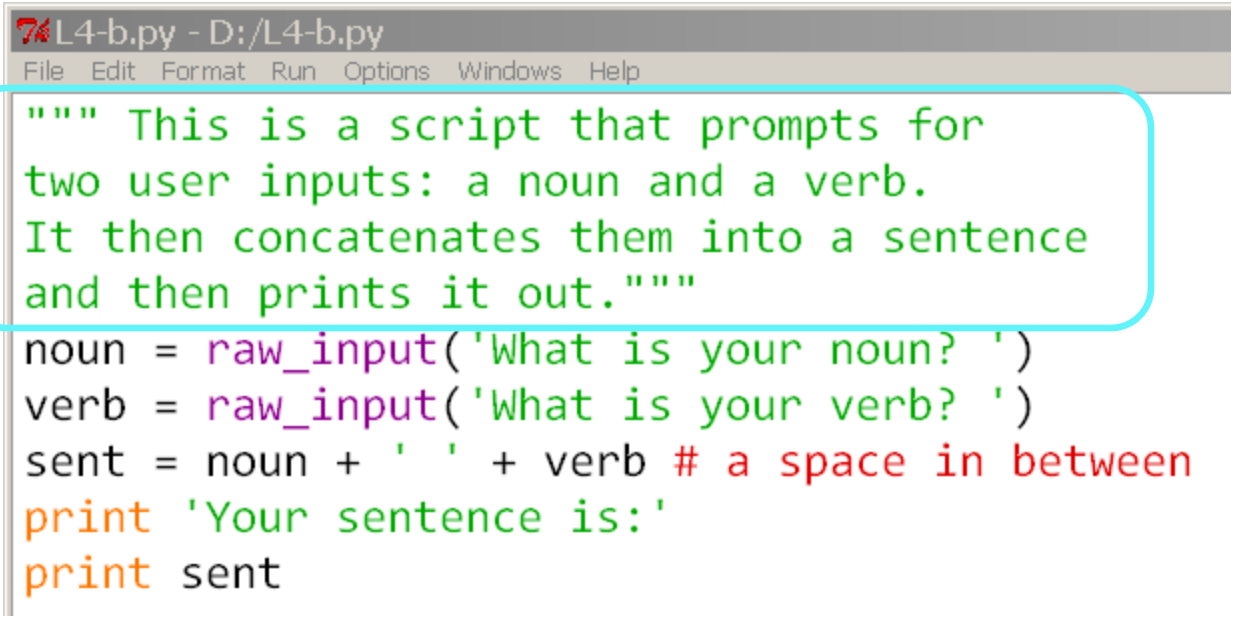
File Edit Format Run Options Windows Help

```
# Takes noun and verb from user input
# and outputs a sentence as a single string
noun = raw_input('What is your noun? ')
verb = raw_input('What is your verb? ')
sent = noun + ' ' + verb # a space in between
print 'Your sentence is:'
print sent
```

  - Good comments make programs more readable and will help you diagnose problems. Get in the habit of commenting!

Multi-line comments

- Multi-line strings (in """ … """) are often used for a long comment spanning across multiple lines:



- NOTE: These multi-line strings are NOT a true comment in that they are *not* ignored by the interpreter.

Scope and indentation

"If it is snowing shovel the sidewalk and do your homework."

- Is it:
  - If it is snowing
    - shovel the sidewalk
    - and do your homework

```
snowing = False
if snowing :
    print 'Bart must shovel sidewalk'
    print 'Bart must do homework'
```

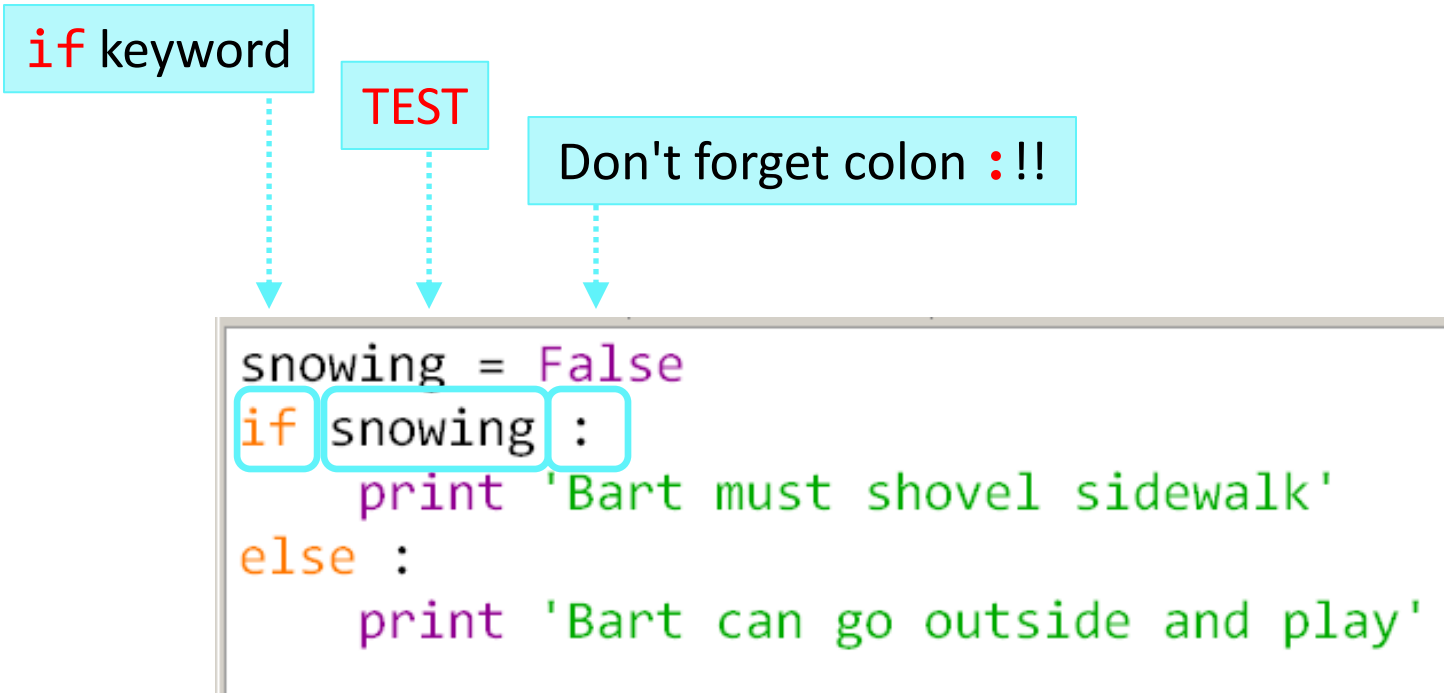
or:

  - If it is snowing
    - shovel the sidewalk
  - and do your homework

```
snowing = False
if snowing :
    print 'Bart must shovel sidewalk'
print 'Bart must do homework'
```

Python uses **indentation** to define code blocks.

Conditional statement

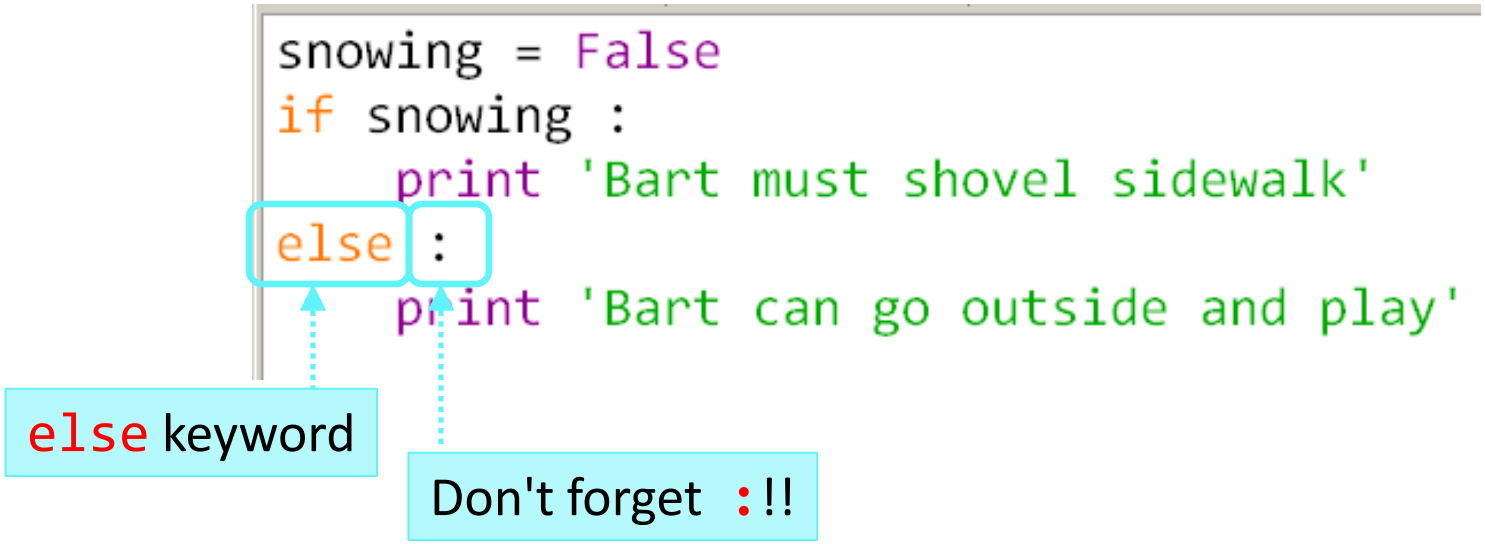


if … else …

Indented block under `if :` Executes only when TEST evaluates to **True**

```
snowing = False
if snowing :
    print 'Bart must shovel sidewalk'
else :
    print 'Bart can go outside and play'
```

if … else …


snowing = False
if snowing :
print 'Bart must shovel sidewalk'
else :
print 'Bart can go outside and play'
else keyword
Don't forget :!!

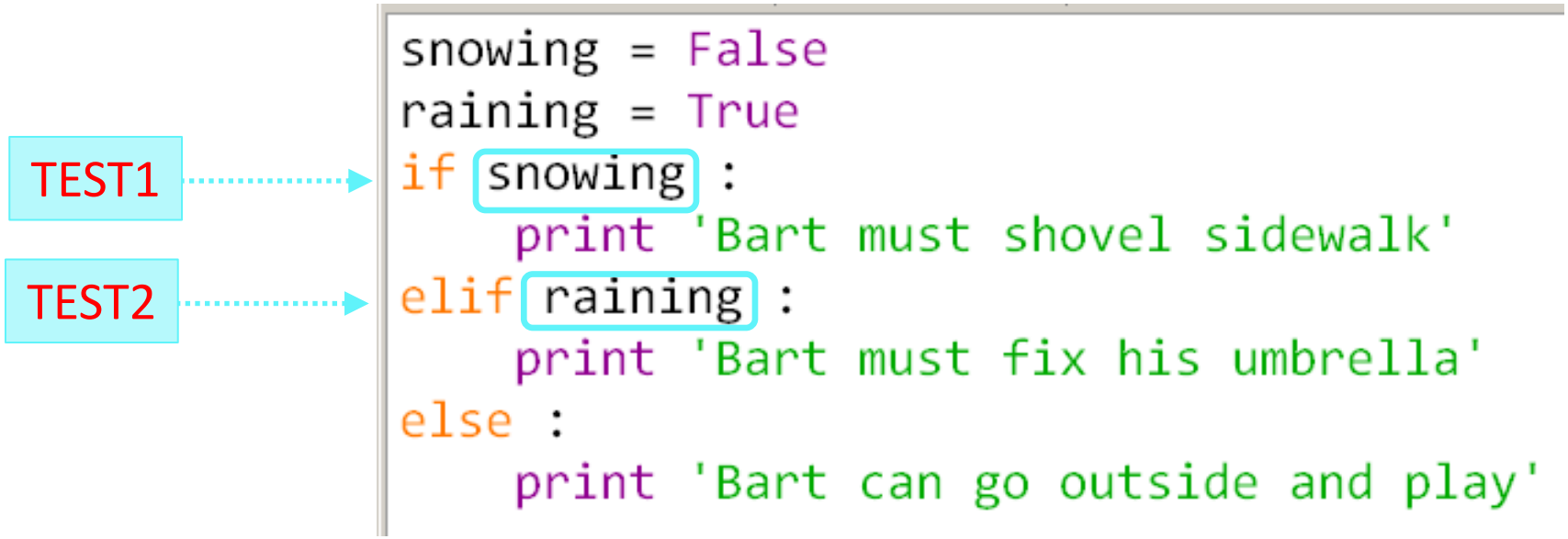
if … else …

```
snowing = False
if snowing :
    print 'Bart must shovel sidewalk'
else :
    print 'Bart can go outside and play'
```

Indented block under `else :` Executes only when TEST evaluates to **False**

if … elif … else …

- `elif` ← "else if"



if … elif … else …

- `elif` ← "else if"

Executes iff TEST1 is **True**

Executes iff TEST1 is **False** AND TEST2 is **True**

Executes iff TEST1 and TEST2 are both **False**

```
snowing = False
raining = True
if snowing :
    print 'Bart must shovel sidewalk'
elif raining :
    print 'Bart must fix his umbrella'
else :
    print 'Bart can go outside and play'
```

if … elif … elif … else …

```
snowing = False
raining = True
windy = True
if snowing :
    print 'Bart must shovel sidewalk'
elif raining :
    print 'Bart must fix his umbrella'
elif windy :
    print 'Bart must close every window'
else :
    print 'Bart can go outside and play'
```

- What will be the output?

Conditional and Boolean values

- `if TEST :`
- *TEST* can be any expression that evaluates to a Boolean value: **True** or **False**.
  - `if True :`
  - `if 'cat'.endswith('at') :`
  - `if 'et' in 'scattered' :`
  - `if mary == mary2 :`
  - `if mary == mary2 or mary == mary3 :`
  - … and many more!

Comparison operations

Expression	Function
==	*equal to*
!=	*not equal to*
<	*less than*
>	*greater than*
<=	*less than or equal to*
>=	*greater than or equal to*

Numeric comparisons

```
>>> 3 == 4
False
>>> 3*4 == 12
True
>>> 3 == 4
False
>>> 3 != 4
True
>>> 9**2 == 81
True
>>> 3 < 4
True
>>> 3 <= 3
True
>>> 3 >= 4
False
```

String comparisons

```
>>> 'coffee' == 'tea'
False
>>> 'coffee' < 'tea'
True
>>> 'Coffee' < 'coffee'
True
>>> 'coffee' < 'Tea'
False
>>> len('coffee') < len('tea')
False
```

Alphabetic ordering
= Unicode character ordering

All uppercase
before lowercase!

String comparisons

```
>>> 'coffee' == 'tea'
False
>>> 'coffee' < 'tea'
True
>>> 'Coffee' < 'coffee'
True
>>> 'coffee' < 'Tea'
False
>>> len('coffee') < len('tea')
False
```

Alphabetic ordering
= Unicode character ordering

All uppercase
before lowercase!

- String comparisons are based on the ordering of character code points in Unicode
- All uppercase letters precede all lowercase letters!

Practice 1

L4-b.py - D:/L4-b.py

File Edit Format Run Options Windows Help

```
# Takes noun and verb from user input
# and outputs a sentence as a single string
noun = raw_input('What is your noun? ')
verb = raw_input('What is your verb? ')
sent = noun + ' ' + verb # a space in between
print 'Your sentence is:'
print sent
```

```
>>>
What is your noun? cats
What is your verb? sleep
Your sentence is:
cats sleep
>>> ========================
>>>
What is your noun? Mary
What is your verb? sing
Your sentence is:
Mary sing
```

sings

- Let's modify our noun-verb script so that the verb has <u>number agreement</u>
  - How to do this? See if noun ends in 's', and if it does not, attach 's' suffix to the verb

    <-- use `if… : else : !`

Practice 1

1 minute

L4-b.py - D:/L4-b.py

File Edit Format Run Options Windows Help

```
# Takes noun and verb from user input
# and outputs a sentence as a single string
noun = raw_input('What is your noun? ')
verb = raw_input('What is your verb? ')

if                          ❶ :          # if noun is plural
    sent = noun + ' ' + verb             # no verbal inflection needed
else :                                     # if noun is not plural
    sent =                          ❷    # then put verb suffix 's'

print 'Your sentence is:'
print sent
```

Practice 1

1 minute

L4-b.py - D:/L4-b.py

File Edit Format Run Options Windows Help

```
# Takes noun and verb from user input
# and outputs a sentence as a single string
noun = raw_input('What is your noun? ')
verb = raw_input('What is your verb? ')

if noun.endswith('s') :          # if noun is plural
    sent = noun + ' ' + verb     # no verbal inflection needed
else :                              # if noun is not plural
    sent = noun + ' ' + verb + 's' # then put verb suffix 's'

print 'Your sentence is:'
print sent
```

Practice 2

- Let's polish up our sentence-generating program a little bit more:

```
>>>
What is your noun? snow
What is your verb? fall
Your sentence is:
"Snow falls."
```

  - Capitalize the first word <-- use `.capitalize()` method
  - Add a period at the end
  - Enclose the sentence in quotation " "

Practice 2

1 minute

L4-b.py - D:/L4-b.py

File Edit Format Run Options Windows Help

```
# Takes noun and verb from user input
# and outputs a sentence as a single string
noun = raw_input('What is your noun? ')
verb = raw_input('What is your verb? ')

if noun.endswith('s') :           # if noun is plural
    sent = noun + ' ' + verb      # no verbal inflection needed
else :                               # if noun is not plural
    sent = noun + ' ' + verb + 's'   # then put verb suffix 's'

print 'Your sentence is:'
print                                 ❶
      # Capitalize the first character, end with period
      # and put quotes " " around the sentence
```

Practice 2

1 minute

L4-b.py - D:/L4-b.py

File Edit Format Run Options Windows Help

```
# Takes noun and verb from user input
# and outputs a sentence as a single string
noun = raw_input('What is your noun? ')
verb = raw_input('What is your verb? ')

if noun.endswith('s') :           # if noun is plural
    sent = noun + ' ' + verb      # no verbal inflection needed
else :                               # if noun is not plural
    sent = noun + ' ' + verb + 's'   # then put verb suffix 's'

print 'Your sentence is:'
print '"' + sent.capitalize() + '."'
      # Capitalize the first character, end with period
      # and put quotes " " around the sentence
```

Practice 3

- But there are plural nouns that do not end with 's':
  - ***men***, ***people***, ***mice***, ***geese***

```
>>>
What is your noun? people
What is your verb? walk
Your sentence is:
"People walks."
>>>
```

  - Let's modify the script further so that we get correct **number agreement** with these nouns.

<== expand if … : to include tests to see <u>if noun equals one of the words above</u>

Practice 3

L4-b.py - D:/L4-b.py

File Edit Format Run Options Windows Help

```
# Takes noun and verb from user input
# and outputs a sentence as a single string
noun = raw_input('What is your noun? ')
verb = raw_input('What is your verb? ')

if noun.endswith('s')                                    ❶ \
                                                  ❷ : # if noun is plural
    sent = noun + ' ' + verb                        # no verbal inflection needed
else :                                              # if noun is not plural
    sent = noun + ' ' + verb + 's'                  # then put verb suffix 's'

print 'Your sentence is:'
print '"' + sent.capitalize() + '."'
      # Capitalize the first character, end with period
      # and put quotes " " around the sentence
```

Use \ to break a very long line!

1 minute

Practice 3

L4-b.py - D:/L4-b.py

File Edit Format Run Options Windows Help

```
# Takes noun and verb from user input
# and outputs a sentence as a single string
noun = raw_input('What is your noun? ')
verb = raw_input('What is your verb? ')

if noun.endswith('s') or noun == 'men' or noun == 'people' \
   or noun == 'mice' or noun == 'geese' : # if noun is plural
    sent = noun + ' ' + verb              # no verbal inflection needed
else :                                    # if noun is not plural
    sent = noun + ' ' + verb + 's'        # then put verb suffix 's'

print 'Your sentence is:'
print '"' + sent.capitalize() + '."'
      # Capitalize the first character, end with period
      # and put quotes " " around the sentence
```

1 minute

Python indentation: CAUTION

- Usually 4 SPACES per block: ␣␣␣␣
- Sometimes a single TAB is used instead: →

- "Intelligent" text editors optimized for Python programming provide indentation help:
  - Auto-indentation after `if… :` or `while… :`
  - Auto-convert 1 tab into 4 spaces
  - Default setting still differs editor to editor; you must pay attention

    ← IDLE, Notepad++ (Win), TextWrangler (Mac), JEdit (any OS)
- Other plain text editors do not!
  - You have to manually indent, keep your indentation CONSISTENT

    ← Notepad & Wordpad (Win), TextEdit (Mac)

IDLE and indentation: CAUTION

- Unfortunately, IDLE shell uses different indentation from IDLE editor
  - IDLE shell uses actual **TAB** character (= size of **8 spaces**)
  - IDLE shell also has the >>> prompt…

This one does not line up – >>>!

TAB (8 spaces) instead of 4 spaces

```
>>> snowing = False
>>> raining = True
>>> if snowing :
        print 'Bart must shovel sidewalk'
elif raining :
        print 'Bart must fix his umbrella'
else:
        print 'Bart can go outside and play'


Bart must fix his umbrella
```

- You CANNOT copy-and-paste indented blocks between IDLE editor and shell

`list()` converts string → list

- Splitting a string into a list of substrings:

```
>>> chomsky = 'colorless green ideas'
>>> chomsky.split()
['colorless', 'green', 'ideas']
>>> chomsky.split('e')
['colorl', 'ss gr', '', 'n id', 'as']
```

- But how to convert a string into a list of characters?
  - `list()`

```
>>> list(chomsky)
['c', 'o', 'l', 'o', 'r', 'l', 'e', 's', 's', ' ', 'g', 'r',
'e', 'e', 'n', ' ', 'i', 'd', 'e', 'a', 's']
>>> list('Hello!')
['H', 'e', 'l', 'l', 'o', '!']
```

.join() operation

- How do you turn a list into a string?
  - .join()

```
>>> kids = ['Bart', 'Lisa', 'Nelson', 'Milhouse']
>>> '-'.join(kids)
'Bart-Lisa-Nelson-Milhouse'
>>> ' and '.join(kids)
'Bart and Lisa and Nelson and Milhouse'
```

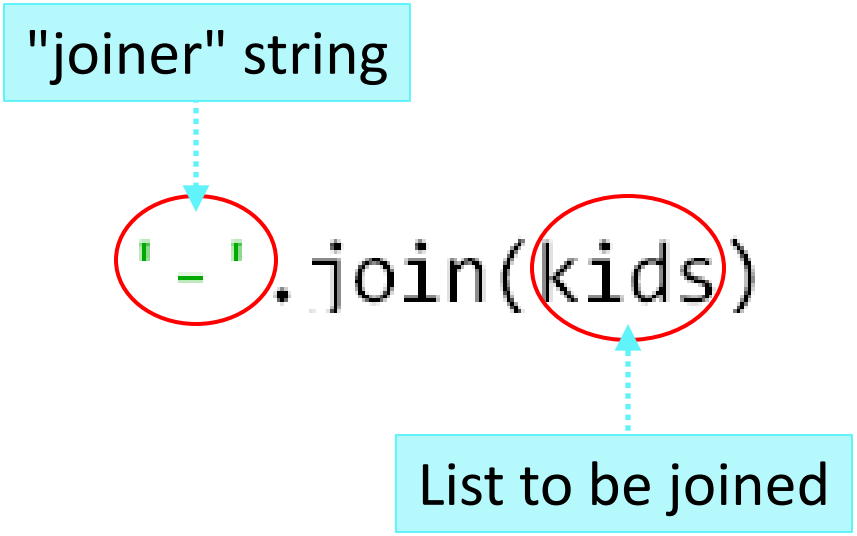
```
>>> creatures = ['man', 'bear', 'pig']
>>> ''.join(creatures)
'manbearpig'
```

.join() syntax

- How do you turn a list into a string?
  - .join()

```
>>> kids = ['Bart', 'Lisa', 'Nelson', 'Milhouse']
>>> '-'.join(kids)
'Bart-Lisa-Nelson-Milhouse'
```

Note the syntax!
.join() method is
called on
the "joiner" string
and not the list



Using .split() and .join() together

- Split a **phrase** (string) into **words** (list), and then join them back with a space

```
>>> mary = 'Mary had a little lamb'
>>> mwords = mary.split()
>>> mwords
['Mary', 'had', 'a', 'little', 'lamb']
>>> ' '.join(mwords)
'Mary had a little lamb'
```

- Split a **word** (string) into **characters** (list), and then join them back with an empty string ''

```
>>> v = 'victory'
>>> list(v)
['v', 'i', 'c', 't', 'o', 'r', 'y']
>>> ''.join(list(v))
'victory'
```

Splitting on **empty string ''** `v.split('')` does NOT work!

Try it out

1 minute

- Split a **phrase** (string) into **words** (list), and then join them back with a space

```
>>> mary = 'Mary had a little lamb'
>>> mwords = mary.split()
>>> mwords
['Mary', 'had', 'a', 'little', 'lamb']
>>> ' '.join(mwords)
'Mary had a little lamb'
```

- Split a **word** (string) into **characters** (list), and then join them back with an empty string ''

```
>>> v = 'victory'
>>> list(v)
['v', 'i', 'c', 't', 'o', 'r', 'y']
>>> ''.join(list(v))
'victory'
```

Reversing a list

- `.reverse()`

List is **MUTABLE**
String is **IMMUTABLE**

```
>>> mwords
['Mary', 'had', 'a', 'little', 'lamb']
>>> mwords.reverse()
>>> mwords
['lamb', 'little', 'a', 'had', 'Mary']
>>> mwords.reverse()
>>> mwords
['Mary', 'had', 'a', 'little', 'lamb']
```

Reverses a list **IN PLACE**: **original list is changed**

```
>>> wd = 'school'
>>> wd.upper()
'SCHOOL'
>>> wd
'school'
```

cf. String operations **do not change the original string**!

Try it out

1 minute

- `.reverse()`

```
>>> mwords
['Mary', 'had', 'a', 'little', 'lamb']
>>> mwords.reverse()
>>> mwords
['lamb', 'little', 'a', 'had', 'Mary']
>>> mwords.reverse()
>>> mwords
['Mary', 'had', 'a', 'little', 'lamb']
```

Reverses a list **IN PLACE**: **original list is changed**

```
>>> wd = 'school'
>>> wd.upper()
'SCHOOL'
>>> wd
'school'
```

cf. String operations **do not change the original string**!

Looking ahead

- `for x in LIST :`
  - iterates through a list for doing something to each element

```
>>> simpsons = ['Homer', 'Marge', 'Bart', 'Lisa', 'Maggie']
>>> for s in simpsons :
        print s, 'is a Simpson.'

Homer is a Simpson.
Marge is a Simpson.
Bart is a Simpson.
Lisa is a Simpson.
Maggie is a Simpson.
>>>
```

Press ↵ **ENTER** at the end of each line

← Iterates through every element `s` of the `simpsons` list and prints the value of `s` followed by 'is a Simpson.' .

Wrap-up

- If you want to try more commands, visit:
  - A Beginner's Python Tutorial, Lesson 4 & 7
    - http://www.sthurlow.com/python/lesson04/
    - http://www.sthurlow.com/python/lesson07/
- Office hours
  - Na-Rae: Mon & Thu 2:30 – 4pm
  - Shameek: Mon 11am – 1pm, **Tue 5-6pm** (changed!)
  - Minas: Mon 5-6pm, Tue 4-5pm
- **Homework #1**
  - http://www.pitt.edu/~naraehan/ling1991/hw1.pdf
  - Due Tuesday midnight
  - **START EARLY!**

Come to PyLing

- PyLing
  - Pitt Python Linguistics Group
- Meets bi-weekly
- Python exercises and guest speakers
- First meeting this semester:
  - **1/27 (Mon) 6:15pm, in our classroom (CL 2818)**
  - There will be pizza!
  - We have a Facebook group