Modules, Scripts, and I/O

Topics:

- Script Mode
- Modules
- The `print` and `input` statements
- Formatting
- First look at importing stuff from other modules

The Windchill Calculation

Let's compute the windchill temperature given that the air temperature is T = 32F and the wind is W = 20mph.

Here is the formula courtesy of the National Weather Service:

$$W_{chill} = (35.74 + \ 0.6215 * T) + (-35.75 + 0.4275 * T) * W^{.16}$$

The formula only applies if T <= 50F and W>=3mph.

Use Python in Interactive Mode

```
>>> Temp = 32
>>> Wind = 20
>>> A = 35.74
>>> B = .6215
>>> C = -35.75
>>> D = .4275
>>> e = .16
>>> WC = (A+B*Temp)+(C+D*Temp)*Wind**e
>>> print WC
19.9855841878
```

The `print` statement is used for displaying values in variables.

Quick Note on `print`

The line

```
>>> print WC
```

results in the display of the value currently housed in the variable `WC`

More on the `print` statement later.

Motivating "Script Mode"

What is the new windchill if the wind is increased from 20mph to 30mph?

Looks like we have to type in the same sequence of statements. Tedious.

Wouldn't it be nice if we could store the sequence of statements in a file and then have Python "run the file" after we changed `Wind = 20` to `Wind = 30` ?

Script Mode

Instead of running Python in interactive mode, we run Python in script mode.

The code to be run (called a script) is entered into a file (called a module).

We then ask Python to "run the script".

What is a Module?

A module is a `.py` file that contains Python code.

In CS 1110, these are created using Komodo Edit.

The Module WindChill.py

WindChill.py

```
Temp = 32
Wind = 20
A = 35.74
B = .6215
C = -35.74
D = .4275
e = .16
WC = (A+B*Temp)+(C+D*Temp)*Wind**e
print WC
```

Running the Module

Here are the steps to follow in the command shell:

1. Navigate the file system so that you are "in" the same diretory that houses **`WindChill.py`**

2. Type: **`python WindChill.py`**

Details

Suppose I have a directory on my desktop called **TODAY** where I keep all my python files for today's lecture.

I navigate the file system until I get this prompt:

```
C:\Users\cv\Desktop\TODAY>
```

Asking Python to Run the Code in WindChill.py

```
C:\Users\cv\Desktop\TODAY> Python WindChill.py

19.6975841877955
```

To save space in subsequent slides, we will refer to
`C:\Users\cv\Desktop\TODAY>` as `Blah\Blah>`

Multiple Statements on a Line

Can put multiple statements on a line. Separate the statements with semicolons.

WindChill.py

```
Temp = 32
Wind = 20
A=35.74;B=.6215;C=-35.74;D=.4275;e=.16
WC = (A+B*Temp)+(C+D*Temp)*Wind**e
print WC
```

For lecture slides I will sometimes do this to save space.
But in general, it makes for ``dense reading'' and should be avoided.

Module Readability: Comments

Comments begin with a "#"

WindChill.py

```
Temp = 32
Wind = 20
# Model Parameters
A=35.74;B=.6215;C=-35.74;D=.4275;e=.16
# Compute and display the windchill
WC = (A+B*Temp)+(C+D*Temp)*Wind**e
print WC
```

Comments: Guidelines

Comments can also appear on the same line as a statement:

```
Wind = 20 # wind speed in miles-per-hour
```

Everything to the right of the "#" is part of the comment and not part of the program.

Comments and Readability

Start each program (script) with a concise description of what it does

Define each important variable/constant

A chunk of code with a specific task should be generally be prefaced with a concise comment.

Module Readability: docstrings

A special comment at the top of the module.

WindChill.py

```
"""Computes windchill as a function of
   wind(mph)and temp (Fahrenheit)."""
Temp = 32
Wind = 20
# Model Parameters
A=35.74;B=.6215;C=-35.74;D=.4275;e=.16
# Compute and display the windchill
WC = (A+B*Temp)+(C+D*Temp)*Wind**e
print WC
```

Docstrings: Guidelines

Docstrings are multiline comments that are delimited by triple quotes: """

They are strategically located at the beginning of "important" code sections.

Their goal is to succinctly describe what the code section is about.

One example of an "important" code section is a module.

Trying Different Inputs

WindChill.py

```
"""Computes windchill as a function of
   wind(mph)and temp (Fahrenheit)."""
Temp = 32
Wind = 20
# Model Parameters
A=35.74;B=.6215;C=-35.74;D=.4275;e=.16
# Compute and display the windchill
WC = (A+B*Temp)+(C+D*Temp)*Wind**e
print WC
```

Can we be more flexible here?

Handy Input

If we want to explore windchill as a function of windspeed and temperature, then it is awkward to proceed by editing the module **`WindChill.py`** every time we want to check out a new wind/temp combination.

The **`input`** statement addresses this issue.

The `input` Statement

The input statement is used to solicit values via the keyboard:

`input`(< *string that serves as a prompt* >)

Later we will learn how to input data from a file.

Temp and Wind via input

WindChill.py

```
"""Computes windchill as a function of
   wind(mph)and temp (Fahrenheit)."""

Temp = input('Enter temp (Fahrenheit):')
Wind = input('Enter wind speed (mph):')

# Model Parameters
A=35.74;B=.6215;C=-35.74;D=.4275;e=.16
# Compute and display the windchill
WC = (A+B*Temp)+(C+D*Temp)*Wind**e
print WC
```

A Sample Run

The prompt is displayed…

```
> Enter temp (Fahrenheit) :
```

And you respond…

```
> Enter temp (Fahrenheit) : 15
```

A Sample Run

The next prompt is displayed…

```
> Enter wind speed (mph) :
```

And you respond again…

```
> Enter wind speed (mph) : 50
```

A Sample Overall "Dialog"

```
BlahBlah> python WindChill.py
Enter temp (Fahrenheit)  : 15
Enter wind speed (mph)   : 50
-9.79781580448
```

For more on Keyboard Input

See the demo file

KeyboardInput.py

It describes another mechanism for acquiring input data called **raw_input.**

More Readable Output

The `print` statement can be used to format output in a way that facilitates the communication of results.

We don't need to show wind chill to the 12th decimal!

More Readable Output

WindChill.py

```
"""Computes windchill as a function of
   wind(mph)and temp (Fahrenheit)."""

Temp = input('Enter temp (Fahrenheit):')
Wind = input('Enter wind speed (mph):')

# Model Parameters
A=35.74;B=.6215;C=-35.74;D=.4275;e=.16
# Compute and display the windchill
WC = (A+B*Temp)+(C+D*Temp)*Wind**e
print '          Windchill :%4.0f' % WC
```

The "Dialog" With Formatting

```
BlahBlah> WindChill
Enter temp (Fahrenheit) :  15
Enter wind speed (mph)  :  50
-9.79781580448
```

print without formatting

```
BlahBlah> WindChill
Enter temp (Fahrenheit) :  15
Enter wind speed (mph)  :  50
              Windchill : -10
```

print with formatting

The `print` Statement

The `print` statement tries to intelligently format the results that it is asked to display.

`print` with formatting puts you in control.

Later we will learn how to direct output to a file

The print Statement

Script:

```
x = 2./5.
print x
x = 1./3.
print x
x = 1234.5678901234
print x
```

Output:

```
0.4
0.333333333333
1234.56789012
```

For float values, print (by itself) displays up to 12 significant digits

The print Statement

Script:

```
x = 1234
y = 12345678
print x,y
```

Output:

```
1234 12345678
```

To display more then one value on a line, separate the references with commas.
A single blank is placed in between the displayed values.

The %f Format

```
x = 1234.123456789
print 'x = %16.3f' %x
print 'x = %16.6f' %x
print 'x = %16.9f' %x
```

```
x =          1234.123
x =       1234.123457
x =    1234.123456789
```

Formatted print statements are developed by "trial and error."

%f … with Left Justification

```
x = 1234.123456789
print 'x = %-16.3f' %x
print 'x = %-16.6f' %x
print 'x = %-16.9f' %x
```

```
x =    1234.123
x =    1234.123457
x =    1234.123456789
```

Formatted print statements are developed by "trial and error."

The %e Format

```
x = 1234.123456789
print 'x = %16.3e' %x
print 'x = %16.6e' %x
print 'x = %16.9e' %x
```

```
x =          1.234e+03
x =      1.234123e+03
x =  1.234123456e+03
```

Formatted print statements are developed by "trial and error."

%e with Left-Justification

```
x = 1234.123456789
print 'x = %-16.3e' %x
print 'x = %-16.6e' %x
print 'x = %-16.9e' %x
```

```
x = 1.234e+03
x = 1.234123e+03
x = 1.234123456e+03
```

Formatted print statements are developed by "trial and error."

The %d Format

```
x = 1234
print 'x = %4d'   %x
print 'x = %7d'   %x
print 'x = %10d'  %x
```

```
x = 1234
x =    1234
x =       1234
```

Formatted print statements are developed by "trial and error."

The %s Format

Script:

```
Band = 'The Beatles'
print '%s in 1964' % Band
```

Output:

```
The Beatles in 1964
```

Strings can be printed too

Formatted Print With More than 1 Source Value

Script:

```
y1 = 1964
y2 = 1971
Band = 'The Beatles'
print '%s in %4d and %4d' % (Band,y1,y2)
```

Need parentheses here.

Output:

```
The Beatles in 1964 and 1971
```

`print` with Formatting

`print` `<` *string with formats* `>` **`%`** `(` `<` *list-of-variables* `>` `)`

A string that includes things like %10.3f. %3d, %8.2e, etc

Comma-separated, e.g., x,y,z. One variable for each format marker in the string. The Parentheses are Required if more than one variable.

For more on Formatted Print

See the demo file

`ShowFormat.py`

Why Program Readability and Style is Important

How we "do business" in commerical, scientific, and engineering settings increasingly relies on software.

Lack of attention to style and substandard documentation promotes error and makes it hard to build on one another's software.

Another Detail

All modules that are submitted for grading should begin with three comments.

WindChill.py

```
# WindChill.py
# Charlse Van Loan (cfv3)
# January 1, 2015

                etc
```

Name of module

Your name and netid

Date

A Final Example

Write a script that solicits the area of a circle and prints out the radius

Preliminary Solution

Radius.py

```
A = input('Enter the circle area: ')
r = sqrt(A/3.14)
print r
```

The Math: solve A = pi*r*r for r.

We Get an Error

```
A = input('Enter the circle area: ')
r = sqrt(A/3.14)
print 'The radius is %6.3f' % r
```

```
 r = sqrt(A/3.14)
NameError: name 'sqrt' is not defined
```

sqrt is NOT a built-in function

Final Solution

Radius.py

```
from math import sqrt
A = input('Enter the circle area: ')
r = sqrt(A/3.14)
print 'The radius is %6.3f' % r
```

We are importing the function **sqrt** from the **math** module.

The Math: solve A = pi*r*r for r.

The Idea Behind `import`

People write useful code and place it in modules that can be accessed by others.

The `import` statement makes this possible.

One thing in the math module is the square root function `sqrt`.

If you want to use it in your module just say

```
from math import sqrt
```

Better Final Solution

Radius.py

```
from math import sqrt,pi
A = input('Enter the circle area: ')
r = sqrt(A/pi)
print 'The radius is %6.3f' % r
```

We are importing the function `sqrt` and the constant `pi` from the `math` module.

Can import more than one thing from a module. Much more on import later.

Sample Run

```
C:\Users\cv\Desktop\TODAY> Python Radius.py
Enter the circle area: 10
The radius is  1.785
```

Terminology

interactive shell

A program that allows the user to type in Python expressions and statements one at a time and evalautes them.

Reference: http://www.cs.cornell.edu/Courses/cs1110/2015sp/materials/definitions.php

Terminology

docstring

A string literal that begins and ends with three quotation marks. Document strings are used to write function specs and are displayed by the help() command.

Reference: http://www.cs.cornell.edu/Courses/cs1110/2015sp/materials/definitions.php

Terminology

module

A file containing global variables, functions, classes and other Python code. The file containing the module must be the same name as the module and must end in ".py" A module is used by either importing it or running it as a script.

Reference: http://www.cs.cornell.edu/Courses/cs1110/2015sp/materials/definitions.php