ESCI 386 – Scientific Programming, Analysis and Visualization with Python

Lesson 5 – Program Control
Interactive Input

• Input from the terminal is handled using the `raw_input()` function

```python
>>> a = raw_input('Enter data: ')  
Enter data: 45
>>> a
'45'
```
Interactive Input (cont.)

- The `raw_input()` function treats all input as a string.

- If we wanted to bring in numerical data we have to convert it using either the `float` or `int` functions.

```python
>>> a = raw_input('Enter data: ')  Enter data: 34.7
>>> a = float(a)
>>> a
34.7
```
Interactive Input (cont.)

• We could do the conversion all on one line

```python
>>> a = float(raw_input('Enter data: '))
```
Input of Multiple Values

• To input multiple values on one line we have to be creative, using the split method for strings.

```python
>>> in_string = raw_input('Enter three numbers: ')  
Enter three numbers: 45.6, -34.2, 9
>>> x, y, z = in_string.split(',',)
>>> x, y, z
('45.6', ' -34.2', ' 9')
```
Code Blocks

• A code block consists of several lines of code that are uniformly indented.

• Code blocks can be used with if, else, elif, for, and while statements, as well as others.

• For this class:
  – My preference is 4 spaces for indents, but you can use any number between 2 and 4
  – Be uniform! Pick an indent number and stick with it.
Conditional Statements

• Conditional statements include the if, then, and elif constructs.
• The form for a simple if statement is

```python
if condition:
    any statements to be executed
    if the condition is true
    go here, all indented
    by the same amount
```
Conditional Statements with else

• If there are also statements to be executed if the condition is not true, then the else statement is used as follows:

```python
if condition:
    any statements to be executed
    if the condition is true
go here, all indented
    by the same amount
else:
    any statement to be executed
    if the condition is false
go here, again all indented
by the same amount
```
Multiple Conditions

- If there are multiple conditions to consider, then the `elif` statement is used:

```python
if condition1:
    any statements to be executed
    if condition1 is true
    go here, all indented
    by the same amount

elif condition2:
    any statements to be executed
    if condition2 is true
    go here, all indented
    by the same amount

elif condition3:
    any statements to be executed
    if condition3 is true
    go here, all indented
    by the same amount

derelationship
    any statement to be executed
    if none of the previous conditions are true
    go here, again all indented
    by the same amount
```
Single-line Conditional Statements

- Python does contain a single line form of an if-else statement. This has the form

  expression1 if condition else expression2

- In this construct, expression1 is executed if condition is True, while expression2 is executed if condition is False.

```python
>>> x = 5
>>> print('Yes') if x <=10 else 'No'
Yes
>>> x = 12
>>> print('Yes') if x <=10 else 'No'
No
```
Loops

• Looping in Python is accomplished using either the for or the while statements.

• The most common way to loop is to use the for statement.

• The for statement requires an iterable object such as a list, a tuple, a range, an array, or even a string.
for Loop

• The basic construct for a for loop is:
for elem in iterable_object:
    statements to be executed within loop.

• For each pass through the loop the next item in the iterable object is passed to the variable elem.

• elem can be any valid variable name.
  – It should be a new variable, not one already used.
for Loop Example

```python
>>> for n in [1, 3, 'hi', False]:
    print(n)
1
3
hi
False
```
for Loop Example

```python
>>> for n in range(-5,30,5):
    print(n)

-5
0
5
10
15
20
25
```
for Loop Example

```python
>>> for n in 'Hello':
    print(n)
Hello
Hello
Hello
```
for Loop Example

```python
>>> b = [(1, 4, 3), (-3, 5, 2), (7, 1, -3)]
>>> for x, y, z in b:
    s = x + y + z
    print(x, y, z, s)
(1, 4, 3, 8)
(-3, 5, 2, 4)
(7, 1, -3, 5)
```
Using enumerate()

- The enumerate() function converts an iterable object into an enumerator object

- This allows the index of the elements to be obtained.

```python
>>> a = [1, 3, 'hi', False]
>>> for i, n in enumerate(a):
    print(i, n)
(0, 1)
(1, 3)
(2, 'hi')
(3, False)
```
while Loops

• The while loop construct will execute the statements within a loop as long as a condition is met. It has the form:

```
while condition:
    statements to be executed
    while the condition remains True
```
>>> a = [1, 3, 4, 5, 'hi', False]
>>> i = 0
>>> while a[i] != 'hi':
    print(a[i])
    i += 1

1
3
4
5
• The continue statement can be used within a loop to skip to the top of the loop.

```python
a = [1, 3, 5, 3, -8, 'hi', -14, 33]
for n in a:
    if n == 'hi':
        continue
    print(n)
```

1
3
5
3
-8
-14
33
Breaking out of a Loop

• The break statement can be used to exit a loop prematurely.

```python
a = [1, 3, 5, 3, -8, 'hi', -14, 33]
for n in a:
    if n == 'hi':
        break
    print(n)
```

```
1
3
5
3
3
-8
```
Verifying Input with while Loop

• A while loop can be used to ensure that interactive input meets certain bounds.

```python
import numpy as np
x = -99
while x < 0:
    x = raw_input('Enter non-negative number: ')  # converts input to floating point
    x = float(x)
print('y = ', np.sqrt(x))
```

Enter non-negative number: -6
Enter non-negative number: -3
Enter non-negative number: 5
('y = ', 2.2360679774997898)