Intro to Python

P. TenHoopen - WMLUG

What is Python?

Python is a cross-platform object-oriented programming language invented by Guido van Rossum. It is an interpreted language but there is support for compiling the programs.

The most recent version is 3.0 (also known as Python 3000) which was released on December 3, 2008. It is incompatible with the 2.x releases.

- Python Home Page - http://www.python.org/
- Beginners Guide - http://wiki.python.org/moin/BeginnersGuide/NonProgrammers
- Python Language Reference - http://docs.python.org/reference/index.html#reference-index
- Guido van Rossum Personal Page - http://www.python.org/~guido/

Interactive Python

Python can be run in interactive mode from the command line. Just type `python` in a terminal.

You will get a >>> prompt.

Commands are entered one line at a time and executed as they are entered. Using the arrow keys, you can scroll up and down to see previously entered commands.

Press `Ctrl-D` to exit.

Python Script Files

Python script files are standard text files, usually saved with a `.py` extension.

On Linux, the first line should contain:

```
#!/usr/bin/python
```

or wherever Python is installed.

Running Python Scripts

Running scripts from the command line:

```
python scriptname.py
```

If the script is executable, you can just use the name:

```
scriptname.py
```

Comments

The # (pound sign) is used to indicate comments and any text following it on that line is ignored.

However, even though the `#!/usr/bin/python` line begins with #, it is not a comment.

Some examples:

```
# This line is ignored by the script processor

a = 1    # this text is an inline comment
```

Variables

Variables are used to hold information. They can contain text (strings) or numbers.

Variables are created by using the equals sign (=). On the left, specify the name of your variable, follow it with an equals sign and give the value.

Examples:

```
# integer variable
a = 1

# string variable

name = "Bart"

# floating point variable

b = 1.5
```

String Indexing

You can index a string to get parts of it. The index starts at 0.

Example:

```
word = "abcd"

first = word[0]      # a
second = word[1]     # b
middle = word[1:3]   # bc
last = word[-1]      # d
```

The -1 index is always the last letter or element.

String Length

The `len` function determines the length of a string.

Example:

```
word = "abcd"
length = len(word)  # 4
```

If Statement

`If` statements test a condition, and if it is true, execute their code block.

Simple `if` statements are in the form of:

```
if condition:
   statement 1
   statement 2
   statement n
```

Everything indented under the `if` statement will be processed if the condition is true.

Important: Python uses indentation to create code blocks as opposed to a language like C that uses braces `{}`.

Note: Don't forget the colon after the condition.

Simple If

Example:

```
a = 1
if a == 1:
   print "a equals 1" # line only runs if a is 1

print "This line runs no matter what 'a' is."
```

Note: To test for equality, == was used. Don't use a single = because that is used for value assignment.

Complex If

`If` statements can use `elif` (else if) and `else`.

Example:

```
a = 1
if a == 1:
   print "a equals 1"
elif a == 2:
   print "a equals 2"
else
   print "a is not 1 or 2"
```

While Loops

`While` loops execute their code block while the condition is true. Once it is false, the program continues with the next non-indented code.

Like the `if` statement, there is a colon after the condition.

Simple `while` loops are in the form of:

```
while condition:
    statement 1
    statement 2
    statement n
```

While Loop Example

Example:

```
a = 3
while a > 0:
    print "Countdown = ", a
    a = a - 1
print "Bang!"
```

This outputs the following to the screen:

```
Countdown = 3
Countdown = 2
Countdown = 1
Bang!
```

Increment/Decrement

Note: The "`a = a - 1`" line can be rewritten in shorthand as "`a -= 1`". Likewise, += increments the variable by the amount indicated.

```
a += 3    # same as: a = a + 3
```

Break Out

You can exit out of a `while` loop if needed by using a `break` statement.

This is useful when needing to exit a loop without having to go through all of the iterations. It is also needed to get out of a continuous loop.

```
while True:      # loop forever
    num = int(raw_input("Enter 0 to quit: "))
    if num == 0:
        break
    print num
```

Continue On

You can use the `continue` statement to skip the rest of a `while` loop's iteration and go back to the condition test.

This is useful when needing to skip parts of a loop depending on another condition.

```
a = 0
while a < 3:
   a += 1
   if a == 2:
      continue  # skip 2
   print a
```

This outputs the following to the screen:

```
1
3
```

While Loop Example 2

```
a = 5
while a > 0:
   if a == 2:
      print "Abort!"
      break
   print "Countdown = ", a
   a -= 1

if a == 1:
   print "Bang!"
```

This outputs the following to the screen:

```
Countdown = 5
Countdown = 4
Countdown = 3
Abort!
```

Python Script Input

Command line Python scripts can get user input using the `raw_input` function.

Examples:

```
name = raw_input("What is your name? ")

number = raw_input("Please enter a number: ")
```

However, the variable `number` will be a string so we need to use the `int` function to convert it to a number:

```
number = int(number)
```

Or, do this instead:

```
number = int(raw_input("Please enter a number: "))
```

Python Script Output

Command line Python scripts can output lines with the `print` command.

Example:

```
print "Hello"
```

The above line outputs the word `Hello` to the screen.

```
name = "Bart"
print "Hello", name
```

The above lines output `Hello Bart` to the screen.

Putting It All Together

```
#!/usr/bin/python

# Python example 1
# P. TenHoopen, 11/18/08

# ask the user for their name
name = raw_input("Please enter your name: ")

# give a welcome message and display the name
print "Hello and welcome", name, "!"
# print a blank line
print

# ask the user for a small number
# make sure to use the int function to convert the raw input
# otherwise you can't use it as a number and numerical tests will fail
times = int(raw_input("Please enter a number between 1 and 10: "))
```

Putting It All Together

```
# check if user entered a zero
if times == 0:
    print "Can't be zero.  Try again to see the 'cool' stuff."
# check if user entered a number greater than 10
# elif is short for "else if"
elif times > 10:
    print "Can't be bigger than 10.  Try again to see the 'cool' stuff."
else:
    # show what the user entered
    print "You entered: ", times
    print

    # set the variable count to 1
    # it will be used to count loop iterations
    count = 1
    # loop while the loop counter is less than or equal to number entered
    while count <= times:
        # show a short message for each loop iteration
        print count, name, "is cool!"
        # increase the loop counter
        count = count + 1
```

Putting It All Together

```
# display a blank line
print
# tell the user how many letters are in their name
print "There are", len(name), "characters in your name."
# show the first character of the name variable
print "The first letter of your name is", name[0]
# show the last character of the name variable
print "The last letter of your name is", name[-1]

print
print "End of program."
```