Introduction to Python Programming

What is Python? Python is a high-level scripting language which can be used for a wide variety of text processing, system administration and internet-related tasks. Unlike many similar languages, it's core language is very small and easy to mas ter, while allowing the addition of modules to perform a virtually limitless variety of tasks. Python is a true object-oriented language, and is available on a wide variety of platforms. There's even a python interpreter written entirely in Java, further enhancing python's position as an excellent solution for internet-based problems.

Python was developed in the early 1990's by Guido van Rossum, then at CWI in Amsterdam, and currently at CNRI in Virginia. In some ways, python grew out of a project to design a computer language which would be easy for beginners to learn, yet would be powerful enough for even advanced users. This heritage is reflected in python's small, clean syntax and the thor oughness of the implementation of ideas like object-oriented programming, without eliminating the ability to program in a more traditional style. So python is an excellent choice as a first programming language without sacri f icing the power and advanced capabilities that users will eventually need.

Although pictures of snakes often appear on python books and websites, the name is derived from Guido van Rossum's favorite TV show, "Monty Python's Flying Circus". For this reason, lots of online and print documen tation for the language has a light and humorous touch. Interestingly, many experienced programmers report that python has brought back a lot of the fun they used to have programming, so van Rossum's inspiration may be well expressed in the language itself.

The very Basics of Python

There are a few features of python which are different than other program ming languages, and which should be mentioned early on so that subsequent examples don't seem confusing. Further information on all of these features will be provided later, when the topics are covered in depth.

Python statements do not need to end with a special character– the python interpreter knows that you are done with an individual statement by the presence of a newline, which will be generated when you press the "Return" key of your keyboard. If a statement spans more than one line, the safest course of action is to use a backslash (\) at the end of the line to let python know that you are going to continue the statement on the next line; you can

continue using backslashes on additional continuation lines. (There are situations where the backslashes are not needed which will be discussed later.)

Python provides you with a certain level of freedom when composing a program, but there are some rules which must always be obeyed. One of these rules, which some people find very surprising, is that python uses in dentation (that is, the amount of white space before the statement itself) to indicate the presence of loops, instead of using delimiters like curly braces ({}) or keywords (like “begin” and “end”) as in many other languages. The amount of indentation you use is not important, but it must be consistent within a given depth of a loop, and statements which are not indented must begin in the first column. Most python programmers prefer to use an edi tor like emacs, which automatically provides consistent indentation; you will probably find it easier to maintain your programs if you use consistent in dentation in every loop, at all depths, and an intelligent editor is very useful in achieving this.

Invoking Python

There are three ways to invoke python, each with its’ own uses. The first way is to type “python” at the shell command prompt. This brings up the python interpreter with a message similar to this one:

```
Python 2.2.1 (#2, Aug 27 2002, 09:01:47)
[GCC 2.95.4 20011002 (Debian prerelease)] on linux2 9
Type "help", "copyright", "credits" or "license" for more information.
```

The three greater-than signs (>>>) represent python’s prompt; you type your commands after the prompt, and hit return for python to execute them. If you’ve typed an executable statement, python will execute it immediately and display the results of the statement on the screen. For example, if I use python’s print statement to print the famous “Hello, world” greeting, I’ll immediately see a response:

```
>>> print ’hello,world’ hello,world
```

The print statement automatically adds a newline at the end of the printed string. This is true regardless of how python is invoked. (You can suppress the newline by following the string to be printed with a comma.)

When using the python interpreter this way, it executes statements im mediately, and, unless the value of an expression is assigned to a variable (See Section 6.1), python will display the value of that expression as soon as it's typed. This makes python a very handy calculator:

```
>>> cost = 27.00
>>> taxrate = .075
>>> cost * taxrate 2.025
>>> 16 + 25 + 92 * 3
317
```

When you use python interactively and wish to use a loop, you must, as always, indent the body of the loop consistently when you type your statements. Python can't execute your statements until the completion of the loop, and as a reminder, it changes its prompt from greater-than signs to periods. Here's a trivial loop that prints each letter of a word on a separate line — notice the change in the prompt, and that python doesn't respond until you enter a completely blank line.

```
>>> word = 'python'
>>> for i in word:
 ... print i
 ...
p
y
t
h
o
n
```

The need for a completely blank line is peculiar to the interactive use of python. In other settings, simply returning to the previous level of indentation informs python that you're closing the loop.