A Comparison of the Basic Syntax of Python and Java

Python

Python supports many (but not all) aspects of object-oriented programming; but it is possible to write a Python program without making any use of OO concepts.

Python is designed to be used interpretively. A Python statement may be entered at the interpreter prompt (>>>), and will be executed immediately. (Implementations make some use of automatic compilation into bytecodes (`.pyc` files).

Python is dynamically typed:

- A variable is introduced by assigning a value to it. Example:

```
someVariable = 42
```

- A variable that has been assigned a value of a given type may later be assigned a value of a different type. Example:

```
someVariable = 42
someVariable = 'Hello, world'
```

Python supports the following built-in data types:

- Plain integers (normally 32-bit integers in the range -2147483648 through 2147483647).
- Long integers (size limited only by memory size of the machine running on)
- Booleans (`False` and `True`).
- Real numbers.
- Complex numbers.

In addition, Python supports a number of types that represent a collection of values - including strings, lists, and dictionaries.

Java

Java supports only object-oriented programming.

Programs written in Java must be explicitly compiled into bytecodes (`.class` files), though an IDE may do this automatically in a way that is transparent to the user. Java does not support direct execution of statements - though there are tools like Dr. Java that support this.

Java is statically typed:

- A variable must be explicitly declared to be of some type before assigning a value to it, though declaration and assignment may be done at the same time. Examples:

```
int someVariable;
int someVariable = 42;
```

- A variable that has been declared to be of a particular type may not be assigned a value of a different type.

Java has two kinds of data types: primitive types and reference types. Java supports the following primitive data types:

- `byte` - 8-bit integers
- `short` - 16-bit integers
- `int` - 32-bit integers
- `long` - 64-bit integers (Java also supports a class `java.math.BigInteger` to represent integers whose size is limited only by memory)
- `float` - 32-bit real numbers.
- `double` - 32-bit real numbers.
- `boolean` - (`false` and `true`).
- `char` - a single character.

In addition, Java supports arrays of any type as the reference types, and the API includes the class `String` and a large number of classes used for collections of values.

A Comparison of the Basic Syntax of Python and Java

Python is line-oriented: statements end at the end of a line unless the line break is explicitly escaped with \. There is no way to put more than one statement on a single line. Examples:

```
this is a statement
this is another statement
this is a long statement that extends over more \
 than one line
```

Statements in Java always end with a semicolon (;). It is possible for a statement to run over more than one line, or to have multiple statements on a single line. Examples:

```
this is a statement;
this is another statement;
this is a long statement that extends over more
 than one line;
a statement; another; another;
```

Python comments begin with # and extend to the end of the line. Example:

```
# This is a comment
A new statement starts here
```

Java has two kinds of comments. A comment beginning with // extend to the end of the line (like Python comments). Comments can also begin with /* and end with */. These can extend over multiple lines or be embedded within a single line. Examples:

```
// This is a comment
A new statement starts here
/* This is also a comment */
/* And this is also a comment, which is
   long enough to require several lines
   to say it. */
Statement starts /* comment */ then continues
```

Python strings can be enclosed in either single or double quotes (' or ""). A character is represented by a string of length 1. Examples:

```
'This is a string'
"This is also a string" # Equivalent
'c' # A string
"c" # An equivalent string
```

Java strings must be enclosed in double quotes (""). A character is a different type of object and is enclosed in single quotes ('). Examples:

```
"This is a String"
'c' // A character, but not a String
```

Python uses the following operators for constructing compound boolean expressions: and, or and not. Example:

```
not(x > 0 and y > 0) or z > 0
```

Java uses the following operators for constructing compound boolean expressions: &&, ||, ! and ^ (meaning exclusive or) Example:

```
! (x > 0 && y > 0) || z > 0 ^ w > 0
```

In Python, the comparison operators (>, <, >=, <=, == and !=) can be applied to numbers, strings, and other types of objects), and compare values in some appropriate way (e.g. numeric order, lexical order) where possible.

In Java, most of the comparison operators (>, <, >=, and <=) can be applied only to primitive types. Two (== and !=) can be applied to any object, but when applied to reference types they test for same (different) object rather than same (different) value.

A Comparison of the Basic Syntax of Python and Java

There is no universally-accepted Python convention for naming classes, variables, functions etc.

By convention, most names in Java use mixed case. Class names begin with an uppercase letter; variable and function names begin with a lowercase letter. Class constants are named using all uppercase letters with underscores. Examples:

```
AClassName
aVariableName
aFunctionName()
A_CLASS_CONSTANT
```

Python definite looping statements have the form `for` *variable* `in` *expression*: Example:

```
for p in pixels:
   something
```

Java has two kinds of definite looping statements. One has the form `for (`*variable* `in` *collection*`)` Example:

```
for (p in pixels)
   something;
```

Python uses the built-in function `range()` with `for` to loop over a range of integers. Examples:

```
for i in range(1, 10)
   something
```
(`i` takes on values 1, 2, 3, 4, 5, 6, 7, 8, 9)

```
for i in range(1, 10, 2)
   something
```
(`i` takes on values 1, 3, 5, 7, 9)

Java uses a different form of the `for` to loop over a range of integers. Examples:

```
for (int i = 1; i < 10; i ++)
   something;
```
(`i` takes on values 1, 2, 3, 4, 5, 6, 7, 8, 9)

```
for (int i = 1; i < 10; i += 2)
   something;
```
(`i` takes on values 1, 3, 5, 7, 9)

Python indefinite looping statements have the form `while` *condition*: Example:

```
while x > 0:
   something
```

Java has two forms of indefinite looping statement. One has the form `while (`*condition*`)` Example:

```
while (x > 0)
   something;
```

The other has the form `do ... while(condition)`. Example:

```
do
     something;
while(x > 0);
```

A Comparison of the Basic Syntax of Python and Java

Python conditional statements have the form `if` *condition*`:` and an optional else part has the form `else:`. The form `elif` *condition*`:` is allowed as an alternative to an `else:` immediately followed by an `if`. Examples:

```
if x < 0:
   something

if x < 0:
   something
else:
   something different

if x < 0:
   something
elif x > 0:
   something different
else:
   yet another thing
```

Java conditional statements have the form `if (`*condition*`)` and an optional else part has the form `else` (no colon) There is no `elif` form - `else if` is used directly. Examples:

```
if (x < 0)
   something;

if (x < 0)
   something;
else
   something different;

if (x < 0)
   something;
else if (x > 0)
   something different;
else
   yet another thing;
```

Java also has another form of conditional statement (`switch`) which has no analog in Python. Example:

```
switch(i)
{
     case 0:
          something 1;
          something 2;
          ...
          break;
     case 1:
          something else;
          break;
     ....
     default:
          yet something else;
}
```

(The individual cases can include any number of statements [shown for the first case only, but possible for any], normally ending with the special statement `break;`)

A Comparison of the Basic Syntax of Python and Java

The scope of a Python conditional or looping statement is denoted by indentation. (If multiple lines are to be included, care must be used to be sure every line is indented identically). Examples:

```
if x < 0:
    do something
do another thing regardless of the value of x

if x < 0:
    do something
    do something else
    do yet a third thing
do another thing regardless of the value of x
```

The scope of a Java conditional or looping statement is normally just the next statement. Indentation is ignored by the compiler (though stylistically it is still highly desirable for the benefit of a human reader). If multiple lines are to be included, the scope must be delimited by curly braces ({ , }). (Optionally, these can be used even if the scope is a single line.) Examples:

```
if (x < 0)
    do something;
do another thing regardless of the value of x;

if (x < 0)
do something;  // Bad style-don't do this!
do another thing regardless of the value of x;

if (x < 0)
{
    do something;
    do something else;
    do yet a third thing;
}
do another thing regardless of the value of x;

if (x < 0)
{
    do something;
}
do another thing regardless of the value of x;
```

A Comparison of the Basic Syntax of Python and Java

A Python function definition has the form

`def` *function-name*`(`*formal-parameter-list*`):`
    body

Example:

```
def disc(a, b, c):
    return b * b - 4 * a * c
```

- If there are no parameters, an empty list (`()`) is used.
- The body is delimited by indentation, and can be any number of lines.
- A function does not have to return a value; if it does not, no `return` statement is required, though `return` without a value (or with `None`) can be used to end the function execution early.

A Python function is called by specifying its name followed by a list of actual parameters (that must be the same length as the list of formal parameters) Example:

```
discr(1, 2, 1)
```

A Java function definition always occurs in the context of some class. It has the form

type function-name`(`*formal-parameter-list*`)`
`{`
    body
`}`

Example:

```
class Solver
{
       ....
       double disc(double a, double b,
                   double c)
       {
              return b * b - 4 * a * c;
       }
       ...
}
```

- The return type of a Java function must always be specified. If it does not return a value, `void` must be used.
- If there are no parameters, an empty list (`()`) is used.
- A type must be specified for each of the parameters.
- The body is always delimited by curly braces (`{` , `}`) even if the body is a single line. Indentation is ignored by the compiler (though stylistically it is still highly desirable for the benefit of a human reader).
- If the function does not return a value (its type is `void`), no `return` statement is required, though `return` without a value can be used to end the function execution early.

A Java function is called by specifying the object or class to which it is to be applied followed by a dot, then its name followed by a list of actual parameters (that must be the same length as the list of formal parameters, with each actual parameter compatible with the declared type of the corresponding formal parameter) Example:

```
solver.discr(1, 2, 1)
```