

I/O

Python has read and print statements as part of the intrinsic language. Java doesn't, which gives you more flexibility and somewhat more awkward constructions.

# Printing to the Screen

This is easy, using the System Output Stream *out*. There are several related methods:

```
System.out.print( String s);
```

```
System.out.println( String s);
```

```
System.out.printf(String s, args );
```

`println( )` appends a newline character `'\n'` to the string. Both `print` and `println` allow you to print objects of any class by calling the object's `toString( )` method. All objects have a `toString( )` method but the default method just returns a `String` version of the object's location in memory, which is fairly useless. Overwrite `toString( )` if you want a useful method.

`System.out.printf( s, args )` is the same as  
`System.out.print( String.format(s, args )`;

Here's how formats work. String `s` is allow to have *placeholders*: `%d` for integer values, `%f` for floats, and `%s` for strings. There should be one arg for each placeholder.

For example,

```
int x = 500;
```

```
String s = String.format( "Give me $%d.", x);
```

makes `s` the string "Give me \$500."

You could say

```
System.out.printf( "Give me $%d.", x );
```

The placeholders can also take *fieldwidths*:

`%5d` says to format the int using at least 5 spaces,  
padded with blanks on the left

`%-5d` is the same, only padded on the right.

`%5s` pads a string with blanks to take at least 5 spaces.

`%7.3f` says to pad the float to 7 spaces, using  
exactly 3 decimal places, as in 123.456

So if variable *player* is "bob" and *total* is 100

```
System.out.printf( "%s wins $%2d\n", player, total);
```

prints "bob wins \$100"

# Input

The main tool for reading data is the Scanner class.

You can construct a scanner to read from a string:

```
Scanner reader = new Scanner( "This is a string.");
```

or from the keyboard

```
Scanner reader = new Scanner(System.in);
```

or from a file called "foobar.txt"

```
Scanner reader = new Scanner( new File("foobar.txt"));
```

Among the Scanner methods are 3 primary ones for reading:

`next()` // returns the next *token* in the input.

// There are methods for specifying tokens.

// The default is a string delimited by white space,

// which you might think of as a "word".

`nextLine()` // returns the next line of input as a String

`nextInt()` // tries to interpret the next token as an

// integer and if that is successful it returns

// that as an int (i.e, as int 25, not String "25").



These methods are supplemented by 3 predicates (functions that return *true* or *false*) that tell you if there is something in the input to read:

hasNext()

hasNextLine()

hasNextInt()

It is common to have input loops that keep going until the user enters a blank line. Here is one way to code that:

```
Scanner reader = new Scanner(System.in);
Boolean done = false;
while (!done) {
    System.out.print( ">>> " );
    String line = reader.nextLine();
    if (line.length() > 0)
        System.out.println(line);
    else
        done = true;
}
```

Here is a complete program that prints file whose name is in args[0].

```
public class FilePrinter {  
  
    public static void main(String[] args) {  
        String fileName = args[0];  
        Scanner scan = null;  
        try {  
            scan = new Scanner(new File(args[0]));  
        } catch(FileNotFoundException e) {  
            System.out.println("Bad file name" );  
            System.exit(-1);  
        }  
        while (scan.hasNextLine()) {  
            String line = scan.nextLine();  
            System.out.println(line);  
        }  
    }  
}
```

# File Output

To write to a file, open a new `PrintStream` object; the result uses the `print ( )`, `println( )`, and `printf( )` methods you are already used to:

```
public static void letterHome() throws FileNotFoundException {  
    PrintStream writer = new PrintStream( new File("foobar.txt"));  
    writer.println("Dear Mom:");  
    int need = 100;  
    writer.printf( "\tPlease send $%d.\n", need);  
    writer.printf( "\t$%d would be better!\n", 2*need);  
    writer.println( "Love, bob");  
    writer.close();  
}
```

This makes the file:

Dear Mom:

    Please send \$100.

    \$200 would be better!

Love, bob