print "hello"
print "world"

To perform these two commands, then loop back:

The overall type is IO (i.e., taken from the last command.

To output a string:

- The goal is to output a character.

- The size of I/O, unlike arithmetic, pure functions.

We call these "commands", or "actions", for we think of them as performing

General I/O functions in Haskell have type IO, where a could be any

I/O in Haskell

Commands That Return Data
The 10 bracket command is similar to Java try-finally:

```java
try {
    // code
} catch (Left e) {
    // catch
} catch (Right e) {
    // catch
} catch (Either e) {
    // catch
}
```

The 10 try command performs your command and catches all exceptions.

More Exception-Handling Commands

```python
def main():
    try:
        # your code
    except Exception as e:
        # handle exception
    finally:
        # cleanup
```

You can also re-throw the exception with:

```python
raise
```

The Boolean command returns true if there is nothing more to read:

```java
while (true) {
    char c = reader.read();
    if (c == '
```
is simply: getChar >> c -> putChar c >> return c

return c

putChar c
do c -> getChar

Example:

{ ... } do \{ ... \} do •

{ ... } do \{ ... \} do •

do \{ \}

The do construct is just syntactic sugar. It is translated to monad operations:

The Magic of do Explained

The IO a type is really a state transformer—a function that maps old state

to new state. Generally, state transformers belong to the Monad class:

The Magic of IO a Explained