Simple Input/Output

Data In/Data Out

- Programs are not very useful if they cannot communicate their results
- Programs also are not terribly useful if they cannot change their controlling parameters to compute different results
  - Note: always assume that there is no such thing as a “one off” program. Odds are very high that somebody will reuse it even if it seems to have a very limited purpose. Make the program read its input values, don’t hard-code them

Files

- We read from or print to files.
- In Unix nearly everything, including the console, is a file. (Stdin, stderr, stdout are all files.)
- For now we will only deal with the standard streams

Reading Input

- We can ask the user to enter data.
- In some cases we can read data in the form of a command-line option

  <interpreter> myprog --option <somevalue>
- The syntax is highly language specific
- We will learn to read from named files later

List-Directed IO

- List-directed IO allows the compiler or interpreter to format the data.
- Input
  - Fortran read from standard input

```
read(*,*) var1, var2, var3
```

Output

- Every language has a way to print its results. Often the word is something like “print” or “write” with various combinations of other syntax.
- For now we will be printing to the console (standard output)
- Neat output is important for legibility, so in many cases we must *format* the output. The syntax here is also highly specific to languages.

List-Directed IO

- Output
  - Fortran write to standard output

```
print *, var1,var2,var3
write(*,*) var1,var2,var3
```

Formatted Input/Output

- Note: as a general rule, avoid formatted input in Fortran since it can lead to errors.
- In Python there are generally more ways to read input so the distinction isn't as important (but it may be necessary to cast).

Edit Descriptors

- The edit descriptor modifies how to output the variables. They are combined into forms like
- RaF.w
- Where R is a repeat count, a is the descriptor, F is the total field width *including* space for +-, and if requested +-e and exponent, and w is the number of digits to the right of the decimal point.
- This is the basic pattern for many languages (some can't do the repeat, many use F.wa)
- Strings and integers take only F (RaF)

Fortran Formatted Output

- You must construct a format string either in the read/write statement or in a format statement.
- Let's just do the format string in the write statement.

```
write(*,'(i5,2x,i6)') i1,i2
```

Common edit descriptors are

I integer

F real (single precision, decimal output)

E real (single precision, exponential output)

G general

D double precision

A character

Fortran Non-Advancing IO

- If we'd like to write to and read from standard input on the same line we can use non-advancing IO:

```
write(*,'(a)',advance='no') "Enter input value:"
read(*,*) value
```

- Must be formatted
  - 'yes' for advance is valid also but is the default.
  - Can be a character variable so you can decide based on conditionals to advance or not