- `ghc`, an optimized compiler for generating native code.
- `ghci`, an interactive interpreter and debugger.
- `runghc`, a program for running Haskell programs as scripts without compilation.

: ?

`ghci` starts with `Prelude>`, being the standard pre-loaded library, and grows longer with each new loaded module or file.

The prompt can be changed with:

```
:set prompt NEWPROMPT
```

Add: `:module + NewModule`
Remove: `module - NewModule`

The abbreviation `:m` also works.

Use parens to get desired association:

```
ghci> (+) 3 5
8
ghci> (^) 3 5
243
```

You nearly always need to surround the negative number with parens to get the desired association.

- The Boolean literals are `True` and `False`. Their type is `Bool`.
- Numbers and other types are not coerced into Boolean interpretations.

- Haskell uses `/=` for "not equal", instead of `!=`.
- Haskell uses `not` for "not", instead of `!`.

```
:info funcname
```

Use `let`.

```
let meaningOfLife = 42
```

`^` for integer exponents.

`**` for floating point exponents.

They are homogeneous and use the common bracket notation. Final commas are not allowed.

```
ghci> [1..5]
[1,2,3,4,5]
ghci> ['a'..'j']
"abcdefghij"
```

`putStr` and `putStrLn`.

A list of characters. `[Char]` and `String` are synonyms.

```
ghci> let lst = ['h', 'i']
ghci> lst
"hi"
ghci> "" == []
True
ghci>
```

Using list operations.

```
ghci> 'a':"bc"
"abc"
ghci> "foo" ++ "bar"
"foobar"
```

```
:set +t

:unset +t
```

A special variable where `ghci` stores the last expression it returned.

Use the `%` operator.

```
ghci> 11 % 29
11%29
it :: Ratio Integer
```

```
:type expr
```

Lines beginning with $--$ are comments.