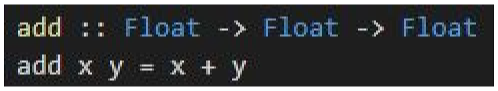
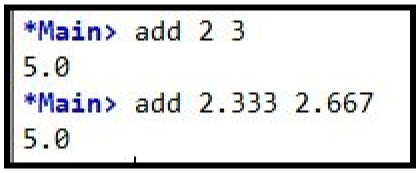
Addition Operator:

- The “+” operator is used for addition.
- E.g.



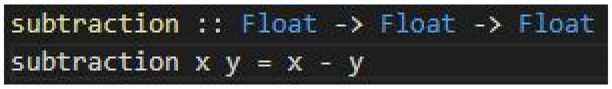
```
add :: Float -> Float -> Float
add x y = x + y
```



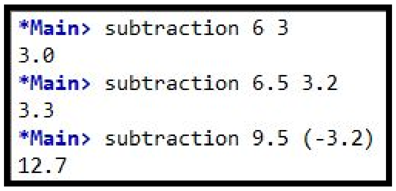
```
*Main> add 2 3
5.0
*Main> add 2.333 2.667
5.0
```

Subtraction Operator:

- The “-” operator is used for subtraction.
- E.g.

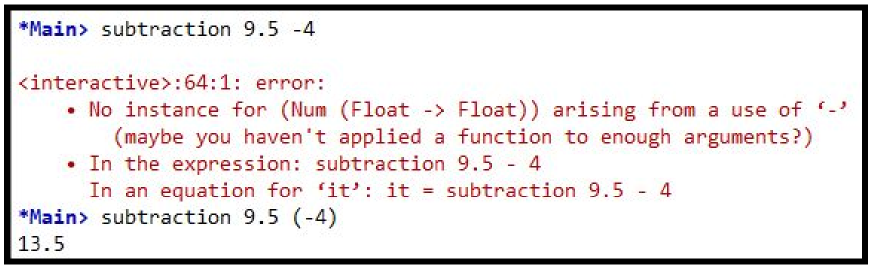


```
subtraction :: Float -> Float -> Float
subtraction x y = x - y
```



```
*Main> subtraction 6 3
3.0
*Main> subtraction 6.5 3.2
3.3
*Main> subtraction 9.5 (-3.2)
12.7
```

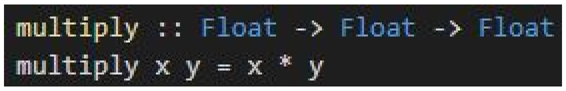
- **Note:** It’s best to use parentheses “()” to enclose negative numbers. Otherwise, the compiler might think the “-” isn’t part of the number.
  E.g.



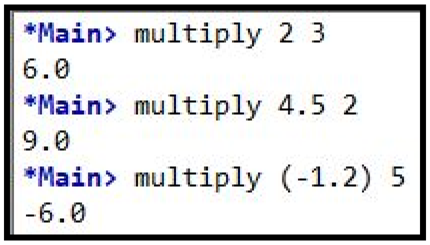
```
*Main> subtraction 9.5 -4

<interactive>:64:1: error:
    • No instance for (Num (Float -> Float)) arising from a use of ‘-’
        (maybe you haven't applied a function to enough arguments?)
    • In the expression: subtraction 9.5 - 4
      In an equation for ‘it’: it = subtraction 9.5 - 4
*Main> subtraction 9.5 (-4)
13.5
```

Multiplication Operator:

- The “*” operator is used for multiplication.
- E.g.



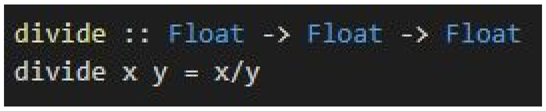
```
multiply :: Float -> Float -> Float
multiply x y = x * y
```



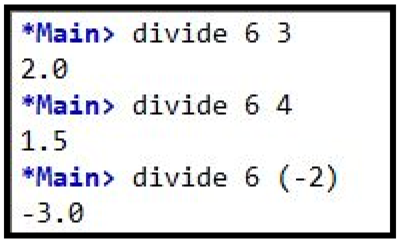
```
*Main> multiply 2 3
6.0
*Main> multiply 4.5 2
9.0
*Main> multiply (-1.2) 5
-6.0
```

Division Operator:

- The “/” operator is used for division.
- E.g.



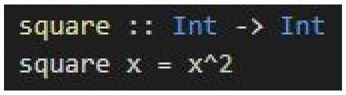
```haskell
divide :: Float -> Float -> Float
divide x y = x/y
```



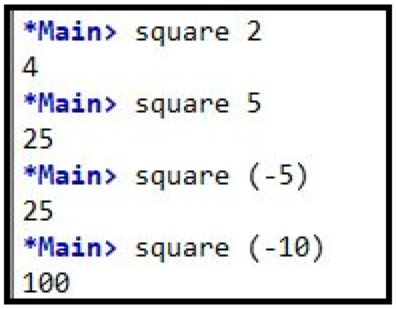
```
*Main> divide 6 3
2.0
*Main> divide 6 4
1.5
*Main> divide 6 (-2)
-3.0
```

Exponent Operator:

- The “^” operator is used for exponent.
- Syntax: **base^exponent**
- E.g.

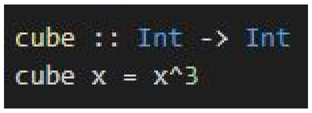


```haskell
square :: Int -> Int
square x = x^2
```

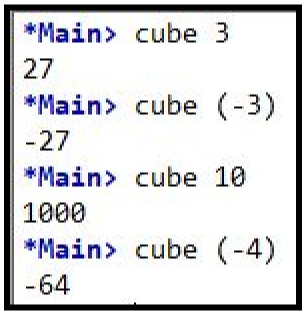


```
*Main> square 2
4
*Main> square 5
25
*Main> square (-5)
25
*Main> square (-10)
100
```

- E.g.



```haskell
cube :: Int -> Int
cube x = x^3
```

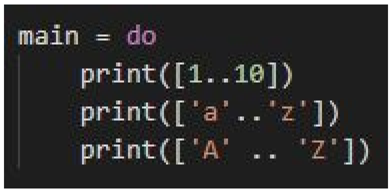


```
*Main> cube 3
27
*Main> cube (-3)
-27
*Main> cube 10
1000
*Main> cube (-4)
-64
```

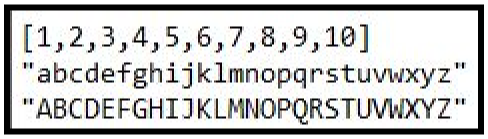
Sequence/Range Operator:

- The “..” operator is used for sequence or range.
- You can use this operator while declaring a list with a sequence of values.
- If you want to print all the values from 1 to 10, then you can use something like "**[1..10]**". Similarly, if you want to generate all the alphabets from "a" to "z", then you can just type "**[a..z]**".

- E.g.



```haskell
main = do
    print([1..10])
    print(['a'..'z'])
    print(['A' .. 'Z'])
```



```
[1,2,3,4,5,6,7,8,9,10]
"abcdefghijklmnopqrstuvwxyz"
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```