# ASGS1115 - Support for Science

Semester 1 - 2019

## LAB week 3: Type and Typeclasses

### List Comprehensions and Tuples

Try running the following expressions:

```
> [5..10]
> [r | r <- [1..5]]
> hask = [h | h <- ["Haskell", "is", "awesome"]]
> odds os = [if (o `mod` 2) == 1 then "odd" else "even" | o <- os]
> addList l1 l2 = [a1+a2 | a1<-l1, a2<-l2]
```

Does anything returns a value that you did not expect? What do you think it is doing?

**Challenges:**

1. Compose a list comprehension that returns even integers between 1 and 50 that is divisible by 5.
2. Define a list comprehension function that returns a list of lists containing all combinations of multiplication of its elements. Such as:

$$\begin{bmatrix} a & b & c \end{bmatrix} \text{ and } \begin{bmatrix} d & e & f \end{bmatrix}) \text{ to } \begin{bmatrix} [ad & ae & ae] \\ [bd & be & bf] \\ [cd & ce & cf] \end{bmatrix}$$

### Tuples

Try running the following expressions:

```
> c1 = (2,5)
> c2 = (1,4)
> cs = (c1, c2)

> c3 = (2,4,6)
> cs = (c1, c2, c3)

> c4 = (5, "five")
> cs = (c1, c2, c4)
```

```
> one = ("one", 1)
> two = ("two", 2)
> five = ("five", 3)
> numstr = [one, two, five]
```

Is there any unexpected behavior from any of the expression? Does that has anything to do with properties of tuples?

Load ListComprehension_and_Tuples.hs script which contains the following functions:

```
-- Alternating letters
alternateLet :: String -> String -> String
alternateLet l "" = ""
alternateLet l (w:ws) = (w:l) ++ (alternateLet l ws)

altLetList l wss = [alternateLet l ws | ws<-wss]

-- Dot product
dotProd2D :: (Double, Double) -> (Double, Double) -> Double
dotProd2D (x1,x2) (y1,y2) = (x1*y1) + (x2*y2)
```

then, try operating these functions to inputs.

**Question:** What do you think these functions are returning corresponding to its inputs?

**Challenge:** Modify the second function so that it can take tuples containing 3 numbers as arguments.

## Types

Try running these expressions in GHCI:

```
> s = "Haskell"
> :t s
> c = 'H'
> :t c
> :t 0.865786
> :t 9
> :t True
> :t ("five", 5)
> :t ("five", 'f')
> :t 3+4+5
> :t True == True
```

What do you think the returned value means? Is there any returned value that you did not expect?

### Type Variables

Load Types_and_Typeclasses.hs script, which contains:

```haskell
-- return first element of a (String, Double) tuple
retf :: (String, Double) -> String
retf (s,n) = s

-- return first element of a tuple with any type of element
retfg :: (a,b) -> a
retfg (x1,x2) = x1
```

Before trying to put in any input to any of these functions, try to read their definitions and understand what each of it will return.

Note that `retf` only take **`String`** and **`Double`**, while `retfg` is a more generalized version of `retf`, where any type of object can be put into the input tuple.

This can be seen in the type of `head` function which returns the first element of a list.

```
> :t head
```

Try this on some of the Haskell built-in functions and note that some of it includes type variables.

**Challenges:** 1. Define a function that reverses a list (of any type). 2. Define a function that reverses a list of 2-tuples (i.e tuple consisting of 2 elements), and swaps the elements of each tuple.

*Input of `[("one", 1), ("two", 2), ("three", 3)]` should be returned as `[(3,"three"),`

## Type Classes

Try assessing some of Haskell's built in operators by running these expressions:

```
> :t (*)
> :t (>)
```

What do you think the returned values mean? How do you think it affects how the operators are used?

Value before the **`=>`** sign is what we call a 'Typeclass' and it restricts the inputs of those operators to types within the specified Typeclass (i.e it restricts the type variable). In a nutshell, Typeclass is a family of objects in Haskell which have one or more common properties.

**Challenge:** List as much type class as you can by identifying types or (using `:t`) of various operators or functions (to list a few: `&&`, `++`, `||`).

**Vocabularies:**

- 2-Tuple
- List Comprehension
- Type
- Type Variable
- Type Class