CS 326 - Functional Programming

Program Design Requirements

Dr. Stephen P. Carl

Designing Programs

We use a specific methodology in all programs written for this class.

- Every function will have a *spec* (sometimes called a *contract*)
- Every function will have a *doc string*
- Every function will have a set of *examples* (optional)
- Every function will have a *definition* (duh!)
- Every function will have runnable *test cases*

Specification

The spec is an annotation showing the types of the arguments the function takes, and the type of data it returns. For example:

```
@spec squares(list(number))::list(number)
```

This says the function **`squares`** takes a list of numbers and returns a list of numbers

The Doc string

The doc string is an annotation that describes what the function is supposed to do. It uses *here-strings,* delimited by """, for multiple lines. For example:

```
@doc """
returns a list made by squaring each element
 in the argument list
"""
```

Now we know what the function is supposed to do.

The Examples

The examples provide an illustrative example of the function in action. It shows the expected output for specific calls to the function, which can also jump-start the process of coming up with test cases.

```
# example: squares [1, 2, 3] -> [1, 4, 9]
# example: squares [a, b, c] -> error
```

Note that by doing some examples, we ensure that we *really* understand what it is the function is supposed to do.

SEWANEE
The University of the South

The Function Definition

The function definition is the sequence of expressions that implements the specification and description.

```
def squares [], do: []
def squares [head | tail], do: [head * head | squares(tail)]
```

SEWANEE
The University of the South

Test Cases

Test cases let us try out the function definition right away. While in examples we'll show just a few tests, in general you should exercise as many different cases as you can think of. Consider *boundary conditions* and error conditions.

```
IO.inspect squares []  # expects []
IO.inspect squares [1 ,2 ,3] # expects [1, 4, 9]
IO.inspect squares [-2 ,0 ,5] # expects [4, 0, 25]
IO.inspect squares [:a ,:b ,"c"] # should be an error
```

SEWANEE
The University of the South

All Together Now

```
@spec squares(list(number))::list(number)
@doc """
returns a list made by squaring each element
"""
def squares [], do: []
def squares [head | tail], do: [head * head | squares(tail)]

IO.inspect squares []         # expects []
IO,inspect squares [-2, 0, 5] # expects [4, 0, 25]
IO.inspect squares [:a, :b, "c"]  # should be an error
```

SEWANEE
The University of the South