The Design Recipe

For the word problems below, assume you have `animalA` and `animalB` defined in your code.

Directions: Define a function called `is-dog`, which consumes a `Row` of the animals table and *computes* whether the animal is a dog.

Contract and Purpose Statement

Every contract has three parts…

```
# is-dog:: (r :: Row) -> Boolean
```

function name | *domain* | *range*

```
# Consumes an animal, and computes whether the species == "dog"
```

what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

```
is-dog("animalA") is animalA["species"] == "dog"
is-dog("animalB") is
```

function name | *input(s)* | *what the function produces*

end

Definition

Write the definition, giving variable names to all your input values…

```
fun is-dog(r):
  r["species"] == "dog"
end
```

function name | *variable(s)* | *what the function does with those variable(s)*

Directions: Define a function called `is-female`, which consumes a Row of the animals table and returns true if the animal is female.

Contract and Purpose Statement

Every contract has three parts…

```
# :: -> 
#
```

function name | *domain* | *range*

what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

```
() is
() is
```

function name | *input(s)* | *what the function produces*

end

Definition

Write the definition, giving variable names to all your input values…

```
fun ():
```

function name | *variable(s)*

what the function does with those variable(s)

end