The Design Recipe

For the word problems below, assume `animalA` and `animalB` are defined as the data rows for Felix and Midnight, respectively.

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

Contract and Purpose Statement

Every contract has three parts...

```
# is-cat:: (r :: Row) -> Boolean
```
function name | *domain* | *range*

```
# Consumes an animal, and computes whether the species == "cat"
```
what does the function do?

Examples

Write some examples, then circle and label what changes...

examples:

```
is-cat ( "animalA" ) is animalA["species"] == "cat"
is-cat ( "animalB" ) is animalB["species"] == "cat"
```
function name | *input(s)* | *what the function produces*

end

Definition

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

```
fun is-cat( r ):
  r["species"] == "cat"
end
```
function name | *variable(s)* | *what the function does with those variable(s)*

Directions: Define a function called `is-young`, which consumes a Row of the animals table and *computes* whether it is less than four years old.

Contract and Purpose Statement

Every contract has three parts...

```
# is-young:: (r :: Row) -> Boolean
```
function name | *domain* | *range*

```
# Consumes an animal, and computes whether the age is less than 4
```
what does the function do?

Examples

Write some examples, then circle and label what changes...

examples:

```
is-young ( "animalA" ) is animalA["age"] < 4
is-young ( "animalB" ) is animalB["age"] < 4
```
function name | *input(s)* | *what the function produces*

end

Definition

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

```
fun is-young( r ):
  r["age"] < 4
end
```
function name | *variable(s)* | *what the function does with those variable(s)*