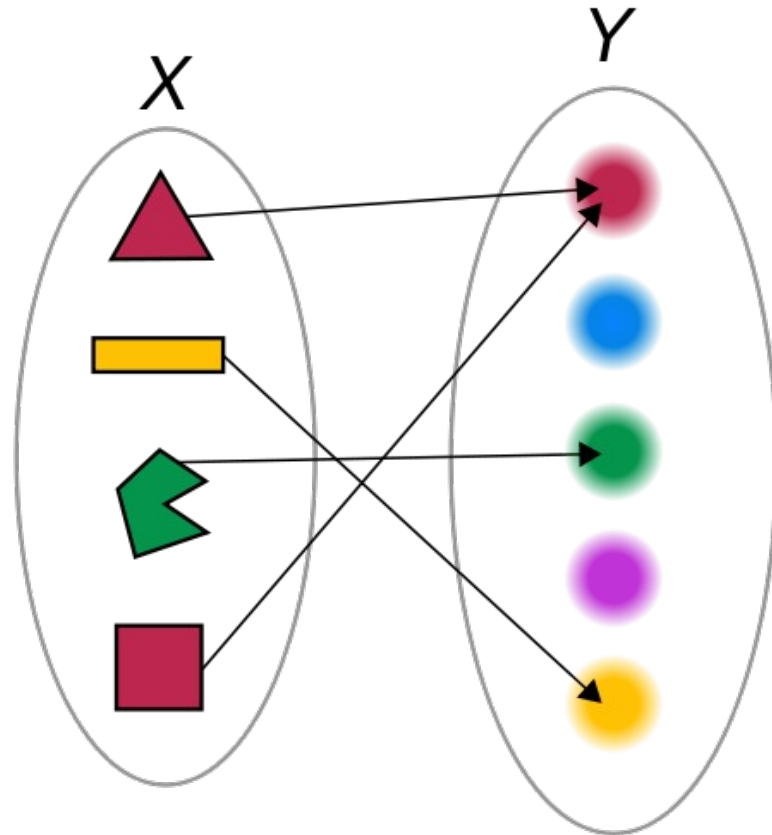




## 4. Writing Functions in R

# Functions

A function generates an output (Y), given an input (X).



# Control Structures: if/else

- Make a logical test
- Perform operations based on the outcome

```
if (condition is true)
{
    # do something
}
```

# Control Structures: if/else

```
age = 21;
```

```
if (age >= 17) {
```

```
    print("You can drive!");
```

```
} else if (age >= 16) {
```

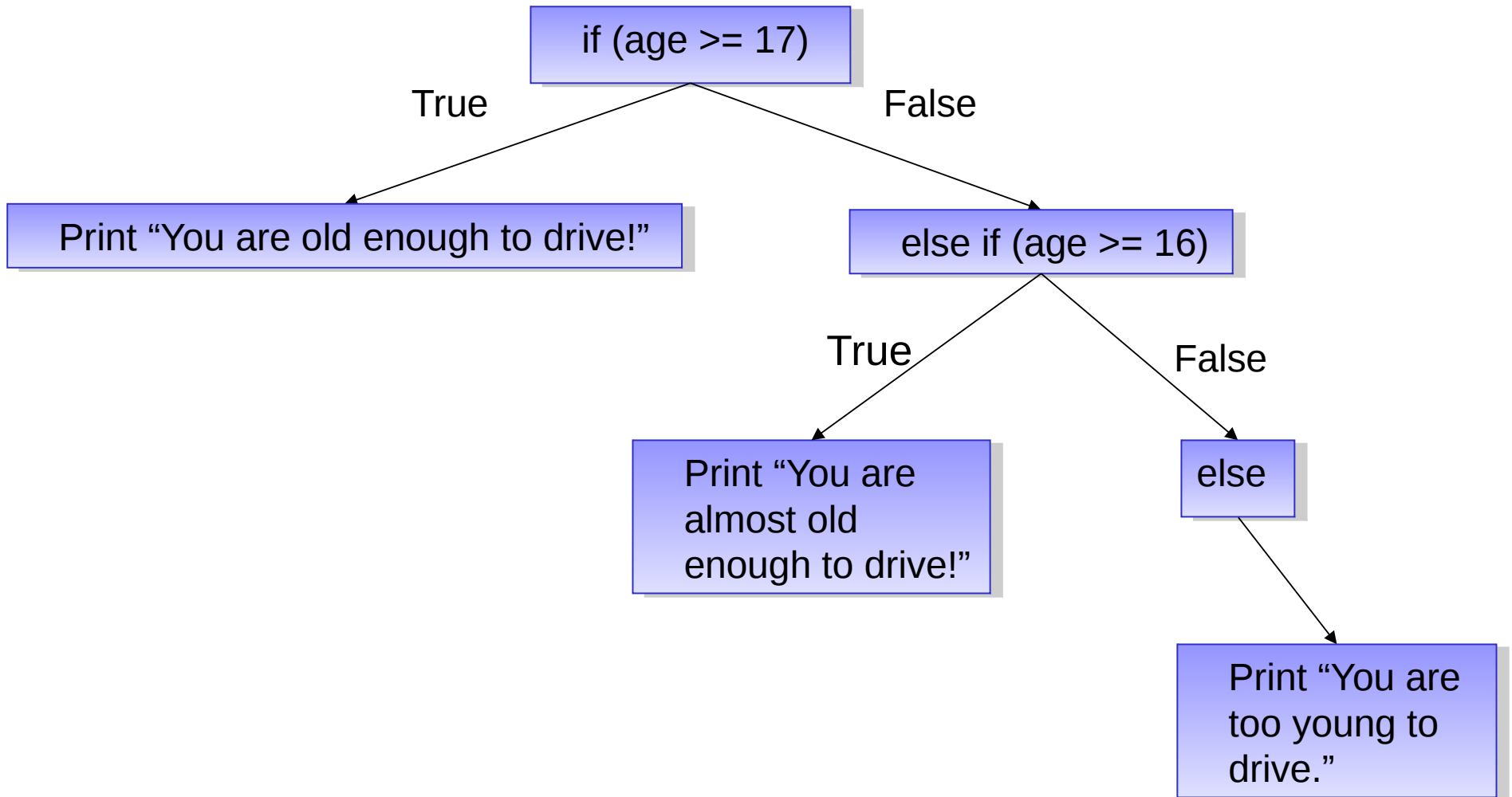
```
    print("You are almost old enough to drive!");
```

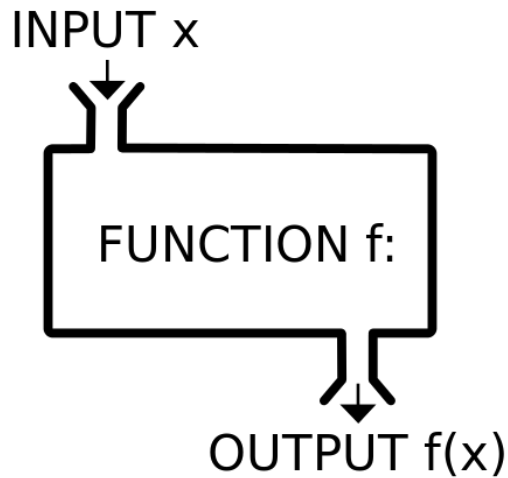
```
} else {
```

```
    print("You are not old enough to drive.");
```

```
}
```

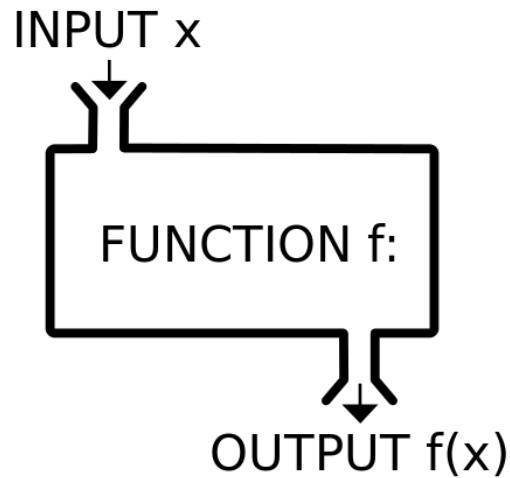
# if/else flowchart





# Functions

- A function  $f$  takes an input,  $x$ , and returns an output  $f(x)$ .
- It's like a machine that converts an input into an output.



# Functions

Function: a piece of code that can be called again and again

To call it, specify:

- Function name
- Input values

It may return an output value

# Functions in R

Name of function

Input parameter(s)

Declaration  
(start of function)

```
functionName = function(inputs) {  
  # do something  
  # return the result  
}
```

End of function

The diagram illustrates the syntax of an R function. It shows the following components: 'functionName' is labeled as the 'Name of function' with an arrow pointing to it; 'inputs' is labeled as the 'Input parameter(s)' with an arrow pointing to it; the opening curly brace '{' is labeled as the 'Declaration (start of function)' with an arrow pointing to it; the closing curly brace '}' is labeled as the 'End of function' with an arrow pointing to it. Between the opening and closing braces, there are two lines of code in green: '# do something' and '# return the result'.



# Functions in R

Name of function

Input parameter(s)

Declaration  
(start of function)

```
toFahrenheit = function(celsius) {  
  f = (9/5) * celsius + 32; # do something  
  return(f); # return the result  
}
```

Output value

End of function

The diagram illustrates the components of an R function definition. The function name 'toFahrenheit' is labeled as the 'Name of function'. The parameter 'celsius' is labeled as the 'Input parameter(s)'. The opening curly brace '{' is labeled as the 'Declaration (start of function)'. The closing curly brace '}' is labeled as the 'End of function'. The expression '(9/5) \* celsius + 32' is labeled as the 'Output value'. The comments '# do something' and '# return the result' are highlighted in green.

# Functions in R

```
toFahrenheit = function(celsius) {  
  f = (9/5) * celsius + 32; # do something  
  return(f); # return the result  
}
```

# Functions in R

```
celsius = c(20:25); # define input temperatures
```

```
toFahrenheit = function(celsius) {  
  f = (9/5) * celsius + 32; # perform the conversion  
  return(f);  
}
```

```
# call the function to convert temperatures to Fahrenheit:
```

```
toFahrenheit(celsius);
```

```
[1] 68.0 69.8 71.6 73.4 75.2 77.0
```

# Control Structures: apply() family

- What if we want to call a function over and over?
- We can do this with a single line of R code!
- Use it on native R functions, or functions you wrote yourself.

```
apply(vector, function)
```

# Control Structures: `sapply()`

```
> lis = c("a", "b", "c", "d")
```

```
> sapply(lis, class)
```

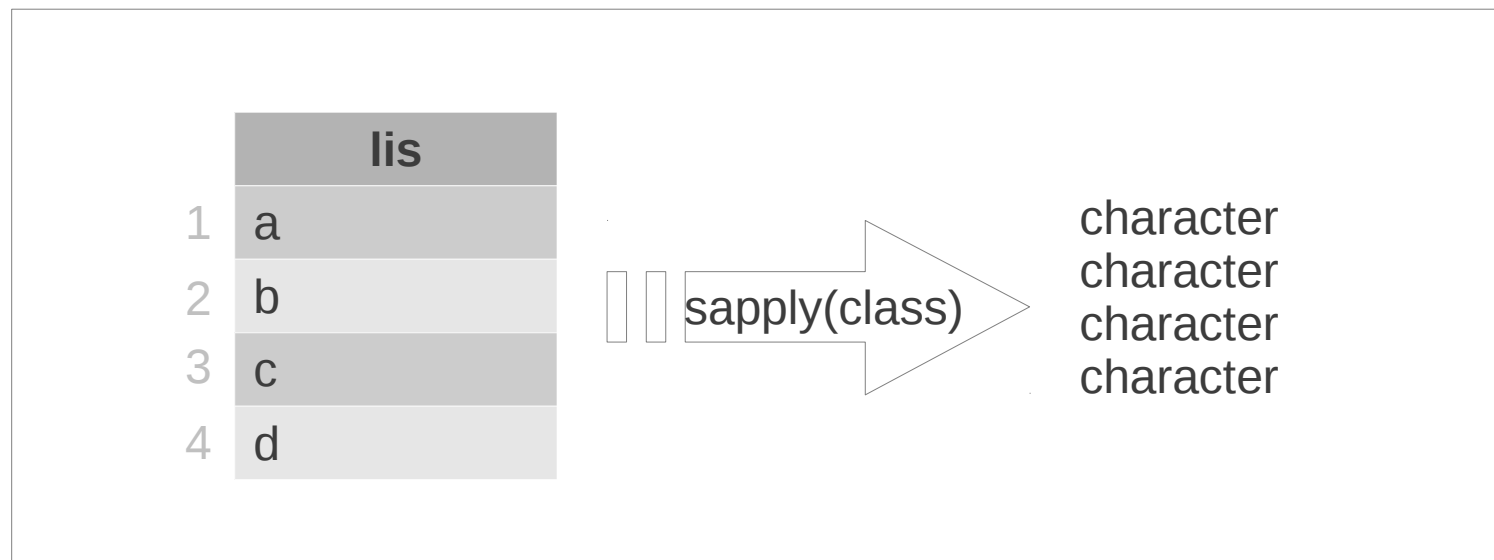
a

b

c

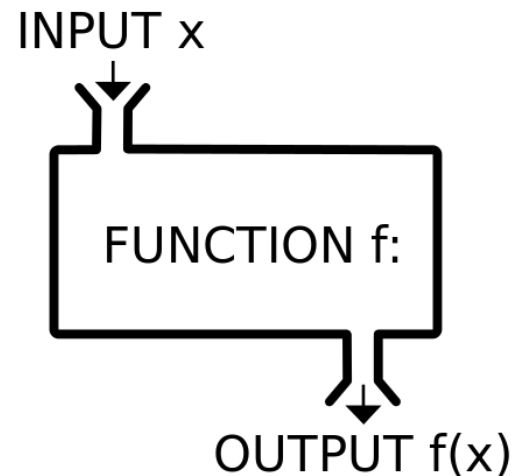
d

"character" "character" "character" "character"



# Tips: Writing Functions

- Use an editor window (not the command line) to compose functions
- Try out one line at a time, and test!
- Comment your function to indicate:
  - input
  - output
  - purpose



# Interlude

Complete function exercises.



Open in the RStudio source editor:

`workshop/exercises/exercises-functions.R`