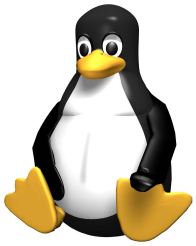
Lab 2: Practice with Penguins



The `penguins` data set in the `{palmerpenguins}` R package will provide you your first data-centric tour between R and Python.

1. Load the data in the R, bring up the helpfile (with `?`) and query its structure (`str`). What is the data structure? What is the observational unit in this data set? How many observations are there and how many variables measured on each?

2. Use `{ggplot2}` to make two scatter plots of bill length as a function of bill depth. For the first, map color to species. For the second, map color to island. Which covariate more cleanly separates the penguins based on bill size?

3. Bring the data into a Python environment using `{reticulate}` and access the `r` object and assign it the name `pypenguins`. What is the `type()` of that data set?

4. You can find the length of an object with `len()`. Try calling that command on `pypenguins`. What is it counting? Try calling the analogous function `length()` on `penguins` in R. Why does it return that value? Isn't `penguins` a rectangular data frame?

5. You can access the keys in `pypenguins` by appending the object with `.keys` and the values by appending with `.values`. If you print them to the console, you'll see the resulting objects are structured like lists. Use the keyword `in` to check if the keys `bill_length_mm`, `bill_depth_mm` and `species` are in `pypenguins`.

6. You can remove elements of a dictionary by using the `del` keyword followed by the element in the dictionary. Use this approach to remove the `year` element, then check that it's been removed by printing out the keys again. This illustrates the mutable nature of dictionaries: you can change them without reassigning them.

7. *List comprehension* is a concise way to filter out particular elements of a list to create a new list. It is analogous to the following logical subsetting of a vector in base R, done to retain only the names of variable that are longer than 8 characters.

```
name_vec <- names(penguins)
name_vec[nchar(name_vec) > 8]
```

```
## [1] "bill_length_mm"    "bill_depth_mm"     "flipper_length_mm"
```

```
## [4] "body_mass_g"
```

In Python, it takes the form:

```
l = list(pypenguins.keys())
[x for x in l if len(x) > 8]
```

```
## ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']
```

Which can be read as: form a new list (`[]`) of all elements `x` from `l` that meet the condition that `len(x) > 8`. We can use the fact that since those objects passing through `x` will all be strings, we can also apply the `.upper()` method on the way out:

```
[x.upper() for x in l if len(x) > 8]
```

```
## ['BILL_LENGTH_MM', 'BILL_DEPTH_MM', 'FLIPPER_LENGTH_MM', 'BODY_MASS_G']
```

Using the fact that each element in `pypenguins` is a list, use list comprehension to create a new list called `short_bills` that contains the bill lengths of all penguins with lengths less than 40 mm. Bring that list back into R and use ggplot2 to create a histogram of its distribution.