ICA8

Harrison Fringer

2023-04-04

Loop Questions

Creating a loop that prints the nth letter of the alphabet in a specified format.

```r
for (o in 1:26){
  print(paste(letters[o], "is", "the", scales::ordinal(o), "letter", "of", "the", "alphabet."))
}
```

```
## [1] "a is the 1st letter of the alphabet."
## [1] "b is the 2nd letter of the alphabet."
## [1] "c is the 3rd letter of the alphabet."
## [1] "d is the 4th letter of the alphabet."
## [1] "e is the 5th letter of the alphabet."
## [1] "f is the 6th letter of the alphabet."
## [1] "g is the 7th letter of the alphabet."
## [1] "h is the 8th letter of the alphabet."
## [1] "i is the 9th letter of the alphabet."
## [1] "j is the 10th letter of the alphabet."
## [1] "k is the 11th letter of the alphabet."
## [1] "l is the 12th letter of the alphabet."
## [1] "m is the 13th letter of the alphabet."
## [1] "n is the 14th letter of the alphabet."
## [1] "o is the 15th letter of the alphabet."
## [1] "p is the 16th letter of the alphabet."
## [1] "q is the 17th letter of the alphabet."
## [1] "r is the 18th letter of the alphabet."
## [1] "s is the 19th letter of the alphabet."
## [1] "t is the 20th letter of the alphabet."
## [1] "u is the 21st letter of the alphabet."
## [1] "v is the 22nd letter of the alphabet."
## [1] "w is the 23rd letter of the alphabet."
## [1] "x is the 24th letter of the alphabet."
## [1] "y is the 25th letter of the alphabet."
## [1] "z is the 26th letter of the alphabet."
```

2. Using a count variable in a while loop that increases count by 1 while it is less than 40. It then checks if this iteration of count is divisible by 4.

```r
count <- 1
while(count < 40){
  count <- (count + 1)
  if (count%%4 == 0){
    next
  }
  print(count)
}
```

```
## [1] 2
## [1] 3
## [1] 5
## [1] 6
## [1] 7
## [1] 9
## [1] 10
## [1] 11
## [1] 13
## [1] 14
## [1] 15
## [1] 17
## [1] 18
## [1] 19
## [1] 21
## [1] 22
## [1] 23
## [1] 25
## [1] 26
## [1] 27
## [1] 29
## [1] 30
## [1] 31
## [1] 33
## [1] 34
## [1] 35
## [1] 37
## [1] 38
## [1] 39
```

```
4%%2
```

```
## [1] 0
```

Vectorized Function Practice

3. Applying the summarize function to each level of the mtcars data set using apply()

```
apply(X = mtcars, MARGIN = 2, FUN = summary)
```

```
##              mpg    cyl     disp       hp     drat      wt     qsec     vs
## Min.    10.40000 4.0000  71.1000  52.0000 2.760000 1.51300 14.50000 0.0000
## 1st Qu. 15.42500 4.0000 120.8250  96.5000 3.080000 2.58125 16.89250 0.0000
## Median  19.20000 6.0000 196.3000 123.0000 3.695000 3.32500 17.71000 0.0000
## Mean    20.09062 6.1875 230.7219 146.6875 3.596563 3.21725 17.84875 0.4375
## 3rd Qu. 22.80000 8.0000 326.0000 180.0000 3.920000 3.61000 18.90000 1.0000
## Max.    33.90000 8.0000 472.0000 335.0000 4.930000 5.42400 22.90000 1.0000
##               am   gear   carb
## Min.    0.00000 3.0000 1.0000
## 1st Qu. 0.00000 3.0000 2.0000
## Median  0.00000 4.0000 2.0000
## Mean    0.40625 3.6875 2.8125
## 3rd Qu. 1.00000 4.0000 4.0000
## Max.    1.00000 5.0000 8.0000
```

4. Using the Iris dataset and the tapply() function to find the mean and standard deviation for each Species of the Sepal.Length column.

```
tapply(X = iris$Sepal.Length, INDEX = as.factor(iris$Species), FUN = mean)
```

```
##     setosa versicolor  virginica
##      5.006      5.936      6.588
```

```
tapply(X = iris$Sepal.Length, INDEX = as.factor(iris$Species), FUN = sd)
```

```
##     setosa versicolor  virginica
##  0.3524897  0.5161711  0.6358796
```

Then checking using the group by and summarize functions to prove they match.

```
iris %>%
  group_by(Species) %>%
  summarize(mean = mean(Sepal.Length), sd = sd(Sepal.Length))
```

```
## # A tibble: 3 x 3
##   Species     mean    sd
##   <fct>      <dbl> <dbl>
## 1 setosa      5.01 0.352
## 2 versicolor  5.94 0.516
## 3 virginica   6.59 0.636
```

Running the test_vec through the chain of ifelse criteria, and printing the value.

```
test_vec <- 100:150
test_vec <- ifelse(test_vec%% 15 == 0, "FizzBuzz",
          ifelse(test_vec%% 5 == 0, "Buzz", ifelse(
            test_vec%% 2 == 0, "Fizz", test_vec)
          ))

print(test_vec)
```

```
##  [1] "Buzz"     "101"      "Fizz"     "103"      "Fizz"     "FizzBuzz"
##  [7] "Fizz"     "107"      "Fizz"     "109"      "Buzz"     "111"
## [13] "Fizz"     "113"      "Fizz"     "Buzz"     "Fizz"     "117"
## [19] "Fizz"     "119"      "FizzBuzz" "121"      "Fizz"     "123"
## [25] "Fizz"     "Buzz"     "Fizz"     "127"      "Fizz"     "129"
## [31] "Buzz"     "131"      "Fizz"     "133"      "Fizz"     "FizzBuzz"
## [37] "Fizz"     "137"      "Fizz"     "139"      "Buzz"     "141"
## [43] "Fizz"     "143"      "Fizz"     "Buzz"     "Fizz"     "147"
## [49] "Fizz"     "149"      "FizzBuzz"
```