functions-prep.Rmd

Azul Carrillo

2022-11-08

Writing our own functions in r

- allows us to understand code without understanding all of the details

```
sum(c(1,2,3))
```

```
## [1] 6
```

```
funtion_name <- function(inputs) {
  output_value <- do_something (inputs)
  return(output_value)
}

{
  a = 2
  b = 3
  a + b

}
```

```
## [1] 5
```

creating a function - define the instructions for a calculation haven't actually used it

```
calc_shrub_vol <- function(length, width, height) {
  area <- length * width
  volume <-  area * height
  return(volume)
}
```

running our function

```
calc_shrub_vol(0.8, 1.6, 2.0 )
```

```
## [1] 2.56
```

```
shrub_vol <-  calc_shrub_vol(0.8, 1.6, 2.0) # storing the output in our function so that we can use it
```

2. how to treat functions as black boxes - function should only know the inputs we pass arguments the program should not Khow anything that goes in the function but the output that the function passes back down to it

```
shrub_vol <-  calc_shrub_vol(0.8, 1.6, 2.0)
```

3. setting different values for arguments

- recreating a function height=1 setting defaults to our functions

```
calc_shrub_vol <- function(length, width, height = 1) {
  area <- length * width
  volume <-  area * height
  return(volume)
}
calc_shrub_vol(0.8, 1.6, 2.0 )
```

```
## [1] 2.56
```

```
calc_shrub_vol(0.8, 1.6) # since we gave a height a default value of 1 when the height is not specifyed
```

```
## [1] 1.28
```

4. When To Use Named And Unnamed Arguments pos based for things that are required named arg things ghat are optional only exeption is when we might get confused

```
calc_shrub_vol(length = 0.8 , width = 1.6, height = 2.0)
```

```
## [1] 2.56
```

```
calc_shrub_vol(height = 2.0, length = 0.8 , width = 1.6) #order doesnt matter when named
```

```
## [1] 2.56
```

```
calc_shrub_vol(0.8 ,  1.6, height = 2.0)
```

```
## [1] 2.56
```

5. combining functions

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
calc_shrub_vol <- function(length, width, height = 1) {
  area <- length * width
  volume <-  area * height
  return(volume)
}

est_shrub_mass <- function(volume) {
  mass <- 2.65 * volume^0.9
  return(mass)
}
#combine the two functions
shrub_volume <- calc_shrub_vol(0.8, 1.6, 2.0)
shrub_mass <- est_shrub_mass(shrub_vol)


#or use pipes
shrub_mass <- calc_shrub_vol(0.8, 1.6, 2.0) %>%
  est_shrub_mass()

#nest functions together can make cose difficult to read
shrub_mass <- est_shrub_mass(calc_shrub_vol(0.8, 1.6, 2.0))
```

6. Calling Functions Inside Of Other Functions

we should never assume that variables from the outter program are avaliable inside of a function but we can allways assume that finctions that are avaliable in the outter program are avaliable to functions to do calculations with

```
calc_shrub_vol <- function(length, width, height = 1) {
  area <- length * width
  volume <-  area * height
  return(volume)
}

est_shrub_mass <- function(volume) {
  mass <- 2.65 * volume^0.9
  return(mass)
}

est_shrub_dim <- function(length, width, height = 1){
  volume <- calc_shrub_vol(length, width, height)
  mass <- est_shrub_mass(volume)
  return(mass)
}
est_shrub_dim(0.8, 1.6, height = 2.0)
```

```
## [1] 6.175354
```

7. tips and tricks navagation menu collapse and expang change settings to diferentiate between functions and variables

```
calc_shrub_vol <- function(length, width, height = 1) {
  area <- length * width
  volume <-  area * height
  return(volume)
}

est_shrub_mass <- function(volume) {
  mass <- 2.65 * volume^0.9
  return(mass)
}

est_shrub_dim <- function(length, width, height = 1){
  volume <- calc_shrub_vol(length, width, height)
  mass <- est_shrub_mass(volume)
  return(mass)
}
```