Creating Functions

The Four Zodiac Elements

THE READER'S DIGEST VERSION

FIRE

Confident
Passionate
Adventurous
Inspired
Independent


Aries


Sagittarius


Leo

EARTH

Grounded
Nurturing
Practical
Level-headed
Sensual


Taurus


Virgo


Capricorn

AIR

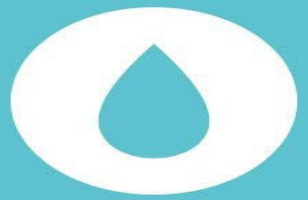
Witty
Social
Free-spirited
Intellectual
Anxious


Gemini


Libra


Aquarius

WATER



Empathetic
Intuitive
Imaginative
Creative
Sentimental


Cancer


Scorpio


Pisces

RD

BMR estimation formulas: The Mifflin St Jeor equation

$$predicted\ calories\ =\ 10 * weight(in\ kg)\ +\ 6.25 * height(in\ cm)\ -\ 5 * age(in\ years)\ +\ s$$

where s is a constant equal to -161 for women and 5 for men.

Function Syntax

Name of the function. Choose something meaningful

The arguments/variables that will take on different values

```
function_name<-function(argument1, argument2,...) {
   Code to perform the task
   return(result)
}
```

After performing the task. It's what the function will return

Example 1: Create a function that computes the predicted number of calories required by women.

BMR estimation formulas: The Mifflin St Jeor equation

$$predicted\ calories\ =\ 10 * weight(in\ kg)\ +\ 6.25 * height(in\ cm)\ -\ 5 * age(in\ years)\ +\ s$$

where s is a constant equal to -161 for women and 5 for men.

Codes:

```
calories_women <- function(weight, height, age){
    calories <- (10 * weight) + (6.25 * height) - (5 * age) - 161
    return(calories)
}
```

What is the output when I run this line after I ran the code that defines my function?

```
calories_women(weight = 65, height = 170, age = 35)
```

Output: 2376.5

calories_women(65,170,35) will get you the same result as

Using default values

```
calories_women_fixed <- function(weight, height, age = 30){
    calories <- (10 * weight) + (6.25 * height) - (5 * age) - 161
    return(calories)
}

calories_women_fixed(weight = 65, height = 170)
calories_women_fixed(weight = 65, height = 170, age = 35)
```

overrides the default values

Example 2: Create a function for which the two arguments are two numbers a and b. Name this function "find_max". Your function should print the maximum value between the two numbers or it should print that the two values are equal in the case that the two numbers are the same.

Code:

```
find_max <- function (a,b){
    if (a>b) {
      return (a)
    }else if (a<b){
      return (b)
    }else if (a==b){
      return("The two numbers are equal")
    }
}
```

Another way:

```
find_max <- function (a,b){
    if (a>b) {
      c<-a
    }else if (a<b) {
      c <-b
    }else if (a==b){
      c<- "The two numbers are equal "
    }
  return(c)
}
```

What would your function print if you ran these codes?

`find_max(1,2)` **Output:** 2

`find_max(5,3)` **Output:** 5

`find_max(6,6)` **Output:** "The two numbers are equal"

Example 3: Create a function that computes the mean of a vector. You have to build this function without the use of the default "mean" function provided by R. You should name this function mean_yourname. Hint: To sum the elements of a vector you can use `sum` and to calculate the number of elements in a vector, you can use `length`.

Codes: See next page

```
mean_abdalla <- function(x){
    my_mean <- sum(x) / length(x)
    return(my_mean)
}
```

What output would you get if you ran these lines:

```
v <- 1:4
mean_abdalla(v)
```

Output: 2.5