



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

MSc in Data & Artificial Intelligence

Tareque Hasnat Tuhin
Mail: 2513674@student.uwtsd.ac.uk

Task 01

```
# 12-1 Simple functions
### Function Definition ###

# Triple - pass by value

triple <- function(x) # Take in a single object x
{
  x<-x*3 # Multiply x by 3
  x # Evaluate the new value of x ready to return
}
|
### Main Script ###

a<-5
triple(a) # Displays return value of triple(a)
a # Leaves original value of a unchanged

b<-c(2.1,5.3,7.9)
triple(b) # Handles floats just as easily as integers
b

c<-c(TRUE,FALSE,TRUE)
triple(c) # Handles Booleans because they are stored as 0,1
c

d<-c("A","B","C")
triple(d) # Fails as R can't do maths with strings
d
```

Environment	History	Connections	Tutorial
  Import Dataset ▾  89 MiB ▾ 			
R ▾  Global Environment ▾			
Values			
a	5		
b	num [1:3] 2.1 5.3 7.9		
c	logi [1:3] TRUE FALSE TRUE		
d	chr [1:3] "A" "B" "C"		
Functions			
triple	function (x) 		

Task 02

```
# 12-2 Exercise 1

### Function Definitions ###

# Function mean_mean(x)

mean_mean <- function(x)
{
  out <- mean(x) # Calculate using mean()
  out # Return out
}

# Function mean_loop(x)

mean_loop <- function(x)
{
  running_total<-0 # Initialise for calculating sum
  element_total<-0 # Initialise for counting number of elements

  for (i in x) # for each element in the vector passed in
  {
    running_total<-running_total+i # Add the element to the running total
    element_total<-element_total+1 # Increment the number of elements
  }
  out<-running_total/element_total # Calculate the mean
  out # Return out
}

# Function mean_function(x)

mean_function <- function(x)
{
  out<-sum(x)/length(x) # Calculate mean using vector functions
  out # Return out
}

### Main script ###

# Create test vectors

int_test<-1:5
float_test<-c(1.2,2.5,3.5,4.5,5.5)

# Test mean_mean

mean_mean(int_test)
mean_mean(float_test)

# Test mean_loop

mean_loop(int_test)
mean_loop(float_test)

# Test mean_function

mean_function(int_test)
mean_function(float_test)
```

Environment	History	Connections	Tutorial
 Import Dataset			
R	Global Environment		
Values			
a	5		
b	num [1:3] 2.1 5.3 7.9		
c	logi [1:3] TRUE FALSE TRUE		
d	chr [1:3] "A" "B" "C"		
float_test	num [1:5] 1.2 2.5 3.5 4.5 5.5		
int_test	int [1:5] 1 2 3 4 5		
Functions			
mean_function	function (x)		
mean_loop	function (x)		
mean_mean	function (x)		
triple	function (x)		

Task 03

```
# 12-3 Mutiple Parameters

### Function Definition ###

# Function raise(x,y)
# Raise x to the power y

raise <- function(x,y) x^y # No brackets needed if a single action

# Function echo_string(n,string)
# Generate a vector containing every element of string n times

echostring <- function(n, string)
{
  return_vector<-NA
  for (i in 1:length(string)) # For each word...
  {
    for (j in 1:n) return_vector<-c(return_vector,
                                   string[i]) # ... add it n times
  }
  return(return_vector[-1]) # Drop the initial NA used to define the vector
}

### Main Script ###

# Testing raise(x,y)

raise(2,3) # Test with values

a<-c(2,3)
raise(a[1],a[2]) # Test with vector elements

b<-c(3,4)
raise(a,b) # Test with vectors

# Testing echostring(n,string)

outstring<-echostring(3,"Echo...") # Test with values
outstring

c<-c("A string","Another string")
outstring<-echostring(3,c)
outstring
```

Environment

History

Connections

Tutorial

Import Dataset

79 MiB

List

R

Global Environment

Values

a

num [1:2] 2 3

b

num [1:2] 3 4

c

chr [1:2] "A string" "Another string"

d

chr [1:3] "A" "B" "C"

float_test

num [1:5] 1.2 2.5 3.5 4.5 5.5

int_test

int [1:5] 1 2 3 4 5

outstring

chr [1:6] "A string" "A string" "A string" "An...

Functions

echostring

function (n, string)

mean_function

function (x)

mean_loop

function (x)

mean_mean

function (x)

raise

function (x, y)

triple

function (x)

Task 04

```
# 12-3 Exercise 2

### Function Definitions ###

# Function build_range(x,y,size)
# x=first value, y=highest last value, size=increment value
# Returns a vector starting at x with increment size,
# with the final value no higher than y

build_range <- function(x,y,size)
{
  outvector<-seq(x,y,by=size)
  return(outvector)
}

# Function print_squares(v, limit)
# v=vector of numerical values, limit=maximum value to be printed
# Prints the squares of the vector elements
# No return object

print_squares <- function(v, limit)
{
  for (i in v) if (i^2 <= limit) print(i^2)
}

# Function print_limit(x,y,size,limit)
# x=first value, y=highest last value,
# size=increment value, limit=max value
# Prints out the squares of every size-th number between x and y
# providing that square is less than limit

print_limit <- function(x,y,size,limit)
{
  numberlist<-build_range(x,y,size) # use build_range to generate
                                     # the input vector for print_squares
  print_squares(numberlist,limit) # call print_squares
}

# The following version is more elegant :)

#print_limit <- function(x,y,size,limit)
#{
#  print_squares(build_range(x,y,size),limit) # function calls function
#}

### Main Script ###

# Test build_range
build_range(4,13,4)

# Test print_squares
print_squares(3:9,40)

# Test print_limit
print_limit(2,9,3,30)

# What happens if we use floats instead of integers?
print_limit(2.1,15.6,2.5,35.2)
```

Environment	History	Connections	Tutorial
   Import Dataset  			
R	Global Environment		
Values			
a	num [1:2]	2 3	
b	num [1:2]	3 4	
c	chr [1:2]	"A string" "Another string"	
d	chr [1:3]	"A" "B" "C"	
float_test	num [1:5]	1.2 2.5 3.5 4.5 5.5	
int_test	int [1:5]	1 2 3 4 5	
outstring	chr [1:6]	"A string" "A string" "A string" "An..."	
Functions			
build_range	function (x, y, size)		
echostring	function (n, string)		
mean_function	function (x)		
mean_loop	function (x)		
mean_mean	function (x)		
print_limit	function (x, y, size, limit)		
print_squares	function (v, limit)		
raise	function (x, y)		
triple	function (x)		