**R Programming**

> #The c() function can be used to create vectors of objects by > #concatenating things together.

> x <- c(0.2, 0.9) ## numeric

# > x

[1] 0.2 0.9

# >

> y <- c(TRUE, FALSE) ## logical

# > y

[1] TRUE FALSE

# >

> z <- c("a", "b", "c") ## character

# > z

[1] "a" "b" "c"

>

> a <- 7:37 ## integer

> a

[1] 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34

[29] 35 36 37

>

> b <- c(1+0i, 2+4i) ## complex

> b

[1] 1+0i 2+4i

>

> c <- c(T, F) ## logical

> c

[1] TRUE FALSE

> d <- c(1.6, "a") ## character

> d

[1] "1.6" "a"

> class(d)

[1] "character"

> typeof(d)

[1] "character"

>

> e <- c(TRUE, 2) ## numeric

> e

[1] 1 2

# >

> f <- c("a", TRUE) ## character

> f

[1] "a" "TRUE"

>

>

> #Objects can be explicitly coerced from one class to another using the as.\* functions, if available > t <- 0:6

> class(t)

[1] "integer"

> as.numeric(t)

[1] 0 1 2 3 4 5 6

> as.logical(t)

[1] FALSE TRUE TRUE TRUE TRUE TRUE TRUE

> as.character(t)

[1] "0" "1" "2" "3" "4" "5" "6"

>

> m <- c("a", "b", "c")

> as.numeric(m)

[1] NA NA NA

Warning message:

NAs introduced by coercion as.logical(m)

NA NA NA

as.complex(m)

NA NA NA

Warning message:

NAs introduced by coercion

> #Add

> n = 1:5 + 6:10

> n

[1] 7 9 11 13 15

>

> c(1, 3, 6, 10, 15) + c(0, 1, 3, 6, 10)

[1] 1 4 9 16 25

> sum(1:5)

[1] 15

> sum(1, 2, 3, 4, 5)

[1] 15

> median(1:5)

[1] 3

>

> c(2, 3, 5, 7, 11, 13) - 2 #subtraction

[1] 0 1 3 5 9 11

>

> -2:2 \* -2:2 #multiplication

[1] 4 1 0 1 4

>

> identical(2 ^ 3, 2 \*\* 3) #we can use ^ or \*\* for exponentiation

[1] TRUE

>

> 1:10 / 3 #floating point division

[1] 0.3333333 0.6666667 1.0000000 1.3333333 1.6666667 2.0000000 2.3333333

2.6666667

[9] 3.0000000 3.3333333

>

> 1:10 %/% 3 #integer division

[1] 0 0 1 1 1 2 2 2 3 3

1:10 %% 3 #remainder after division

[1] 1 2 0 1 2 0 1 2 0 1

>

> l <- c(0, Inf, -Inf, NaN, NA)

> is.finite(l)

[1] TRUE FALSE FALSE FALSE FALSE

>

> (p <- 1:10 >= 5)

[1] FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE

> !p

[1] TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE

>

>

> (q <- 1:10 %% 2 == 0)

[1] FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE

> p&q

[1] FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE TRUE

> p | q

[1] FALSE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

>

> #Classes

> class(c(TRUE, FALSE))

[1] "logical"

> class(sqrt(1:10))

[1] "numeric"

>

> sqrt(1:11)

[1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427

3.000000

[10] 3.162278 3.316625

> class(3 + 1i) #"i" creates imaginary components of complex numbers

[1] "complex"

>

> class(7) #although this is a whole number, it has class numeric

[1] "numeric" class(0.5:4.5) "numeric" class(1:5) #unless all its values are whole numbers

"integer"

>

>

> # is(pattern = "^is", baseenv())

> is.character("red lorry, yellow lorry")

[1] TRUE

> is.logical(FALSE)

[1] TRUE

> is.list(list(a = 1, b = 2))

[1] TRUE

> is.numeric(0.1)

[1] TRUE

> is.integer(0.88)

[1] FALSE

> is.integer(1)

[1] FALSE

> is.double(1)

[1] TRUE

>

> h <- "123.456"

> as(h, "numeric")

[1] 123.456

> as.numeric(h)

[1] 123.456

>

> # Sequences

> seq.int(3, 12) #same as 3:12

[1] 3 4 5 6 7 8 9 10 11 12

> seq.int(3, 12, 2)

[1] 3 5 7 9 11

> seq.int(0.1, 0.01, -0.01)

[1] 0.10 0.09 0.08 0.07 0.06 0.05 0.04 0.03 0.02 0.01

>

# Lengths

length(1:5)

[1] 5

> length(c(TRUE, FALSE, NA))

[1] 3

>

> sn <- c("Sheena", "leads", "Sheila", "needs")

> length(sn)

[1] 4

> nchar(sn)

[1] 6 5 6 5

>

> c(apple = 1, banana = 2, "kiwi fruit" = 3, 4) apple banana kiwi fruit

1 2 3 4

> k <- 1:4

> k

[1] 1 2 3 4

> names(k) <- c("apple", "bananas", "kiwi fruit",

+ "")

> k apple bananas kiwi fruit

1 2 3 4

> names(k)

[1] "apple" "bananas" "kiwi fruit" ""

> names(1:4)

NULL

>

> # Indexing Vectors

> j <- (1:5) ^ 2

> j

[1] 1 4 9 16 25

> j[c(1, 3, 5)]

[1] 1 9 25

> class(j)

[1] "numeric" j

1 4 9 16 25

j[c(-2, -4)]

1 9 25

> j[c(TRUE, FALSE, TRUE, FALSE, TRUE)]

[1] 1 9 25

> j[c(1, NA, 5)] ??????

+ j[c(TRUE, FALSE, NA, FALSE, TRUE)] ?????

+ # Tricky

+ j[-1.9]

Contacting Delphi...the oracle is unavailable.

We apologize for any inconvenience.

> which(j > 10)

[1] 4 5

> which.min(j)

[1] 1

>

> # Vector Recycling

> 1:5 + 1

[1] 2 3 4 5 6

> 1 + 1:5

[1] 2 3 4 5 6

> 1:5 + 1:15

[1] 2 4 6 8 10 7 9 11 13 15 12 14 16 18 20

>

> #Repetition

> rep(1:5, 3)

[1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

> rep(1:5, each = 3)

[1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5

> rep(1:5, times = 1:5)

[1] 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5

> rep(1:5, length.out = 7)

[1] 1 2 3 4 5 1 2

>

> # matrices

m <- matrix(nrow = 2, ncol = 3)

> m

[,1] [,2] [,3]

[1,] NA NA NA

[2,] NA NA NA

>

> dim(m)

[1] 2 3

> m <- matrix(1:6, nrow = 2, ncol = 3)

> m

[,1] [,2] [,3]

[1,] 1 3 5

[2,] 2 4 6

> m <- 1:10

> m

[1] 1 2 3 4 5 6 7 8 9 10

> dim(m) <- c(2, 5)

>

> u <- 1:3

> w <- 10:12

> cbind(u, w) u w

[1,] 1 10

[2,] 2 11

[3,] 3 12

> rbind(u, w)

[,1] [,2] [,3] u 1 2 3 w 10 11 12