R Data Types

Data Types

In programming, data type is an important concept. Variables can store data of different types, and different types can do different things. In R, variables do not need to be declared with any particular type, and can even change type after they have been set:

Example:

```
> my_var <- 30 # my_var is type of numeric
> my_var <- "Sally" # my_var is now of type character (aka
string)
> my_var
[1] "Sally"
> # look, it got the last data type assigned to it
```

Basic Data Types

Basic data types in R can be divided into the following types:

- **numeric** - (10.5, 55, 787)
- **integer** - (1L, 55L, 100L, where the letter "L" declares this as an integer)
- **complex** - (9 + 3i, where "i" is the imaginary part)
- **character** (a.k.a. string) - ("k", "R is exciting", "FALSE", "11.5")
- **logical** (a.k.a. boolean) - (TRUE or FALSE)

We can use the **class()** function to check the data type of a variable:

Example:

```
> # numeric
> x <- 10.5
> class(x)
[1] "numeric"
> # integer
> x <- 1000L
> class(x)
[1] "integer"

> # complex
> x <- 9i + 3
```

```
> class(x)
[1] "complex"

> # character/string
> x <- "R is exciting"
> class(x)
[1] "character"

> # logical/Boolean
> x <- TRUE
> class(x)
[1] "logical"
```

Type Conversion

You can convert from one type to another with the following functions:

- as.numeric()
- as.integer()
- as.complex()
- as.character()

Example

```
> x <- 1L # integer
> y <- 2.2 # numeric
> x  # print value of x
[1] 1
> y  # print value of y
[1] 2.2
> class(x) # specify the class of x
[1] "integer"
> class(y) # specify the class of y
[1] "numeric"
>
> # convert from integer to numeric:
> x <- as.numeric(x)
> # convert from numeric to integer:
> y <- as.integer(y)
> # print values of x and y
> x
[1] 1
> y
[1] 2
> # print the class name of x and y
> class(x)
[1] "numeric"
> class(y)
[1] "integer"
```

R Math

Simple Math

In R, you can use **operators** to perform common mathematical operations on numbers.

Example:

```
> 10+5 # Adding
[1] 15
> 10-5 # Subtraction
[1] 5
> 10*5 # Multiplication
[1] 50
> 10/5 # Division
[1] 2
> 10^5 # Exponentiation
[1] 1e+05 # this mean (1×10^5)
> 10**5 # Exponentiation
[1] 1e+05
```

Built-in Math Functions

R also has many built-in math functions that allows you to perform mathematical tasks on numbers.

For example, the min() and max() functions can be used to find the lowest or highest number in a set:

Example:

```
> max(5, 10, 15)
[1] 15
> min(5, 10, 15)
[1] 5
```

Math operators and built-in functions of R

Category	Operator/Function	Description
Math Operators	+	Addition
Math Operators	-	Subtraction
Math Operators	*	Multiplication
Math Operators	/	Division
Math Operators	^ or **	Exponentiation (e.g., 2^3 or 2**3 for 2 raised to the power of 3)
Math Operators	%%	Modulus (remainder after division, e.g., 5 %% 2 returns 1)
Math Operators	%/%	Integer division (e.g. 5 %/% 2 returns 2)
Math Operators	==	Equality check
Math Operators	!=	Inequality check
Math Operators	<	Less than
Math Operators	>	Greater than
Math Operators	<=	Less than or equal to
Math Operators	>=	Greater than or equal to
Math Operators	&	Logical AND
Math Operators	|	Logical OR
Math Operators	!	Logical NOT
Math Functions	sqrt(x)	Square root of x
Math Functions	abs(x)	Absolute value of x
Math Functions	exp(x)	Exponential of x (e raised to the power of x)
Math Functions	log(x)	Natural logarithm of x
Math Functions	log10(x)	Base-10 logarithm of x
Math Functions	log2(x)	Base-2 logarithm of x
Math Functions	sin(x), cos(x), tan(x)	Trigonometric functions (input in radians)
Math Functions	asin(x), acos(x), atan(x)	Inverse trigonometric functions
Math Functions	sum(x)	Sum of all elements in x
Math Functions	prod(x)	Product of all elements in x (Multiplies all the numbers together)
Math Functions	mean(x)	Mean (average) of x
Math Functions	median(x)	Median of x
Math Functions	sd(x)	Standard deviation of x

Math Functions	var(x)	Variance of x
Math Functions	min(x)	Minimum value in x
Math Functions	max(x)	Maximum value in x
Math Functions	range(x)	Range (min and max) of x
Math Functions	round(x, digits)	Round x to the specified number of digits if x=1.2345 round(x, digits=2) the output will be 1.23
Math Functions	floor(x)	Round down to the nearest integer
Math Functions	ceiling(x)	Round up to the nearest integer
Math Functions	factorial(x)	Factorial of x
Math Functions	choose(n, k)	Number of combinations (binomial coefficient)

Example: Basic math operators:

```
> # Define two numbers
> a <- 10
> b <- 3
>
> # Addition
> addition <- a + b
> print(paste("Addition:", addition))
[1] "Addition: 13"
>
> # Subtraction
> subtraction <- a - b
> print(paste("Subtraction:", subtraction))
[1] "Subtraction: 7"
>
> # Multiplication
> multiplication <- a * b
> print(paste("Multiplication:", multiplication))
[1] "Multiplication: 30"
>
> # Division
> division <- a / b
> print(paste("Division:", division))
[1] "Division: 3.33333333333333"
>
> # Exponentiation
> exponentiation <- a ^ b
> print(paste("Exponentiation:", exponentiation))
[1] "Exponentiation: 1000"
>
> # Modulo
> modulo <- a %% b
> print(paste("Modulo:", modulo))
[1] "Modulo: 1"
>
> # Integer Division
> integer_division <- a %/% b
> print(paste("Integer Division:", integer_division))
[1] "Integer Division: 3"
```

Example: How to use built-in functions in R:

```
> # Example: Using Built-in Math Functions in R
>
> # Define a numeric vector
> numbers <- c(1, 2, 3, 4, 5)
>
> # Sum of elements
> sum_numbers <- sum(numbers)
> print(paste("Sum of elements:", sum_numbers))
[1] "Sum of elements: 15"
>
> # Mean (average) of elements
> mean_numbers <- mean(numbers)
> print(paste("Mean of elements:", mean_numbers))
[1] "Mean of elements: 3"
>
> # Median of elements
> median_numbers <- median(numbers)
> print(paste("Median of elements:", median_numbers))
[1] "Median of elements: 3"
>
> # Standard deviation
> sd_numbers <- sd(numbers)
> print(paste("Standard deviation:", sd_numbers))
[1] "Standard deviation: 1.58113883008419"
>
> # Variance
> var_numbers <- var(numbers)
> print(paste("Variance:", var_numbers))
[1] "Variance: 2.5"
>
> # Minimum value
> min_value <- min(numbers)
> print(paste("Minimum value:", min_value))
[1] "Minimum value: 1"
>
> # Maximum value
> max_value <- max(numbers)
> print(paste("Maximum value:", max_value))
[1] "Maximum value: 5"
>
> # Range (minimum and maximum)
> range_values <- range(numbers)
> print(paste("Range of values:", paste(range_values, collapse =
" - ")))
[1] "Range of values: 1 - 5"
>
> # Absolute value of each element
> abs_numbers <- abs(c(-1, -2, -3, -4, -5))
> print(paste("Absolute values:", paste(abs_numbers, collapse = "
, ")))
```

```
[1] "Absolute values: 1, 2, 3, 4, 5"
>
> # Square root of each element
> sqrt_numbers <- sqrt(numbers)
> print(paste("Square roots:", paste(round(sqrt_numbers, 2), coll
apse = ", ")))
[1] "Square roots: 1, 1.41, 1.73, 2, 2.24"
>
> # Exponentiation (e^x) of each element
> exp_numbers <- exp(numbers)
> print(paste("Exponentiation (e^x):", paste(round(exp_numbers, 2
), collapse = ", ")))
[1] "Exponentiation (e^x): 2.72, 7.39, 20.09, 54.6, 148.41"
>
> # Natural logarithm (ln) of each element
> log_numbers <- log(numbers)
> print(paste("Natural logarithm (ln):", paste(round(log_numbers,
2), collapse = ", ")))
[1] "Natural logarithm (ln): 0, 0.69, 1.1, 1.39, 1.61"
>
> # Base-10 logarithm of each element
> log10_numbers <- log10(numbers)
> print(paste("Base-10 logarithm:", paste(round(log10_numbers, 2)
, collapse = ", ")))
[1] "Base-10 logarithm: 0, 0.3, 0.48, 0.6, 0.7"
```

Note: In R, the `collapse` parameter is used in functions like `paste()` to combine multiple elements of a vector into a single string, with a specified separator between each element. The separator is defined by the value of the `collapse` parameter.

Here's an example to demonstrate its usage:

```
> # Define a character vector
> words <- c("apple", "banana", "cherry")
>
> # Combine the elements into a single string with ", " as
the separator
> combined string <- paste(words, collapse = ", ")
> print(combined string)
[1] "apple, banana, cherry"
```

==========

Problem Questions

1. **Variable Types:**
   - Q1: Create a numeric variable num with a value of 10.5 and print its class.
   - Q2: Create a character variable text with the value "Hello, R!" and print its class.
2. **Arithmetic Operators:**

- Q3: Define two numeric variables a and b with values 8 and 3, respectively. Calculate and print the sum, difference, product, and quotient.
- Q4: Using the variables a and b from Q3, calculate the exponentiation (a^b) and modulo (a %% b) and print the results.

3. **Comparison Operators:**
   - Q5: Define two numeric variables x and y with values 5 and 10, respectively. Use comparison operators to check if x is greater than y, less than y, equal to y, and not equal to y. Print the results.
   - Q6: Using the variables x and y from Q5, use the logical operators & (AND) and | (OR) to combine multiple comparison conditions and print the results.
4. **Assignment Operators:**
   - Q7: Assign the value 25 to a variable z using the standard <- assignment operator. Then, assign the value 30 to the same variable using the = assignment operator. Print the value of z.
5. **Logical Operators:**
   - Q8: Create two logical variables logical1 and logical2 with the values TRUE and FALSE, respectively. Use the logical operators && (AND) and || (OR) to combine these variables and print the results.
6. **String Operators:**
   - Q9: Define two character variables firstName and lastName with values "John" and "Doe". Concatenate these variables to create a full name and print it.