CORE S119 FA24 Prog 1

Grusha Prasad

2024-09-03

Variables

Variables store values. In the following case, we are storing the value "hello" in the variable x and "class" in variable y. The variable gets created when you run the code. You can see all the variables you created in the Values pane on the top right of your window.

```
x = "hello"
y = "world"
```

Code Q1 Create a variable called class_name and store "Truth through the Science Looking Glass" in the variable. Remember you have to run the code for the variable to get created!

```
## WRITE YOUR CODE BELOW THIS LINE
```

Code Q2 Multiply seven times 6, and store the value of this in a variable called life.

```
## WRITE YOUR CODE BELOW THIS LINE
```

You can retrieve the value stored in the variable by typing that variable in a code chunk. For example, if you put the variable in the print() function, it will print the value that is stored in the variable.

```
print(x)
## [1] "hello"
print(y)
## [1] "world"
```

Code Q3 Print the values that are stored in the new variables you created.

```
## WRITE YOUR CODE BELOW THIS LINE
```

You can also change the values in variables by assigning the variable to a new value.

```
x = "hi"
y = "hello"
```

Written Q1 What do you think is stored in x and y now?

Answer

Code Q4 Verify your answer by writing some code.

```
## WRITE YOUR CODE BELOW THIS LINE
```

Data types

Consider the following print statements that look very similar.

```
# case 1
print(42)
## [1] 42
print(is.numeric(42))
## [1] TRUE
print(is.character(42))
## [1] FALSE

# case 2
print("42")
## [1] "42"
print(is.numeric("42"))
## [1] FALSE
print(is.character("42"))
## [1] TRUE
```

Written Q2 What do you think is.numeric() tells you?

Answer:

Written Q3 What do you think is.character() tells you?

Answer:

Written Q4 What is the difference between case1 and case2?

Answer:

You can put values of the same type together in a collection.

```
print(c(1,2,3))
## [1] 1 2 3
print(c(1:10)) ## Note this is a convenient way of getting 10 numbers without typing them all
##  [1]  1  2  3  4  5  6  7  8  9 10

print(c("hi", "hello"))
## [1] "hi"    "hello"
```

#### Code Q5 Create a collection with the numbers 1 to 20 and assign it to a variable called twenty.

```
## WRITE YOUR CODE BELOW THIS LINE
```

Functions

Functions take in inputs, perform some operations on the input, and generate an output.

Written Q5 We have already seen three functions. One of them print(), which prints the input that you pass in. What are the other two functions?

Answer

Different functions take in inputs of different types. For example consider the functions

```
x = c(1:10)

print(sum(x))
## [1] 55
print(mean(x))
## [1] 5.5
print(max(x))
## [1] 10
print(min(x))
## [1] 1
```

If you try to run these same functions on a collection with characters, some of them will not work because these functions require numeric inputs.

```
y = c('a', 'b', 'c')

print(sum(y))
## Error in sum(y): invalid 'type' (character) of argument
print(mean(y))
## Warning in mean.default(y): argument is not numeric or logical: returning NA
## [1] NA
print(max(y))
## [1] "c"
print(min(y))
## [1] "a"
```

Functions can also take in more than one input. For example, paste takes in multiple inputs.

```
x = paste("CORES", 119, "FY")
y = paste("CORES", 119, "FY", sep = "")
z = paste("CORES", 119, "FY", sep = ",")

print(x)
## [1] "CORES 119 FY"
print(y)
## [1] "CORES119FY"
print(z)
## [1] "CORES,119,FY"
```

Written question 6 What do you think paste does? What is the sep = " " or sep = "," is doing?

Answer

Conditionals

Often you will want to do one thing if some condition is met, if not, do some other thing. Here is an example.

```
num = 7

if(num > 10){
  greater_than_10 = TRUE
  print("Number is greater than 10")
} else{
  greater_than_10 = FALSE
  print("Number is less than 10")
}
## [1] "Number is less than 10"
```

Written question 7 The code above will print the wrong thing in one case. Identify the case and explain why it is wrong.

Answer

If you just wanted to assign the variable greater_than_10 without printing, here is a more compact way of doing it.

```

greater_than_10 = ifelse(num > 10, TRUE, FALSE)
print(greater_than_10)
## [1] FALSE
```

You can also have multiple conditionals.

```
score1 = 100
score2 = 80
best = ''

if(score1 > score2){
  best = 'score1'
} else if(score1 < score2){
  best = 'score2'
} else{
  best = 'equal'
}

print(best)
## [1] "score1"
```

As before, you can do the same thing in a more compact way

```
best = ifelse(score1 > score2, 'score1',
              ifelse(score1 < score2, 'score2', 'equal'))
```

Putting it together (Homework)

Write code that does the following:

1. Create a variable called temperature and store the current temperature. Include a comment to indicate whether it is in Celsius or Fahrenheit.
2. Create a variable called hot and store what you would consider to be a hot temperature.
3. Create a variable called cold and store what you would consider to be a cold temperature.
4. If the current temperature is greater than or equal to what you consider hot, then print "It is hot!". If it is less than or equal to what you consider cold, then print "It is cold!". Otherwise print "It is pleasant!".
5. Create a collection of temperatures from the past 5 hours.
6. Compute the average temperature over the past five hours and store it in a variable called avg.
7. If the average temperature is greater than or equal to what you consider hot, then print "The past five hours have been hot!". If it is less than or equal to what you consider cold, then print "The past five hours have been cold!". Otherwise print "The past five hours have been pleasant!".