```
# Harvardx: PH125.1x - (1) Data Science: R Basics
# SECTION 2: VECTORS & SORTING
# ASSESSMENTS


# # # ASSESSMENT 2.1: VECTORS

# # NUMERIC VECTORS

# Here is an example creating a numeric vector named cost
[redacted]
# Create a numeric vector to store the temperatures listed in the
 instructions into a vector named temp
# Make sure to follow the same order in the instructions
[redacted]


# # CHARACTER VECTORS

# here is an example of how to create a character vector
[redacted]

# Create a character vector called city to store the city names
# Make sure to follow the same order as in the instructions
[redacted]


# # CONNECTING NUMERIC AND CHARACTER VECTORS

# Associate the cost values with its corresponding food item
[redacted]

# You already wrote this code
[redacted]

# Associate the temperature values with its corresponding city
[redacted]


# # SUBSETTING VECTORS
```

```
# cost of the last 3 items in our food list:
[redacted]

# temperatures of the first three cities in the list:
[redacted]

# Access the cost of pizza and pasta from our food list
[redacted]

# Define temp
[redacted]

# Access the temperatures of Paris and San Juan
[redacted]


# # SEQUENCES

# Create a vector m of integers that starts at 32 and ends at 99.
[redacted]

# Determine the length of object m.
[redacted]

# Create a vector x of integers that starts at 12 and ends at 73.
[redacted]
# Determine the length of object x.
[redacted]

# Create a vector with the multiples of 7, smaller than 50.
[redacted]

# Create a vector containing all the positive odd numbers smaller
 than 100.
# The numbers should be in ascending order
[redacted]


# # SEQUENCES & LENGTH

# We can create a vector with the multiples of 7, smaller than 50
 like this
[redacted]
```

```
# But note that the second argument does not need to be the last
 number
# It simply determines the maximum value permitted
# so the following line of code produces the same vector as seq(7,
 49, 7)
[redacted]

# Create a sequence of numbers from 6 to 55, with 4/7 increments and
 determine its length

[redacted]


# # SEQUENCES OF CERTAIN LENGTH

# Store the sequence in the object a
[redacted]

# Determine the class of a
[redacted]


# # INTEGERS

# Store the sequence in the object a
[redacted]

# Determine the class of a
[redacted]


# # INTEGERS AND NUMERICS

# Check the class of 1, assigned to the object a
[redacted]

# Confirm the class of 1L is integer
[redacted]


# # COERCION

# Define the vector x
[redacted]
```

```
# Note that the x is character vector
[redacted]

# Typecast the vector to get an integer vector
# You will get a warning but that is ok
[redacted]



# # # ASSESSMENT 2.2: SORTING

# # SORT

# Access the `state` variable and store it in an object
[redacted]

# Sort the object alphabetically and redefine the object
[redacted]

# Report the first alphabetical value
[redacted]

# Access population values from the dataset and store it in pop
[redacted]
# Sort the object and save it in the same object
[redacted]
# Report the smallest population size
[redacted]


# # NEW CODES

# Find the index of the smallest value for variable total
[redacted]

# Find the index of the smallest value for population
[redacted]


# # USING THE OUTPUT OF ORDER

# Define the variable i to be the index of the smallest state
[redacted]

# Define variable states to hold the states
[redacted]
```

```
# Use the index you just defined to find the state with the smallest
[redacted]


# # RANKS

# Store temperatures in an object
[redacted]

# Store city names in an object
[redacted]

# Create data frame with city names and temperature
[redacted]

# Define a variable states to be the state names
[redacted]

# Define a variable ranks to determine the population size ranks
[redacted]

# Create a data frame my_df with the state name and its rank
[redacted]


# # DATA FRAMES, RANKS & ORDERS

# Define a variable states to be the state names from the murders
 data frame
[redacted]

# Define a variable ranks to determine the population size ranks
[redacted]

# Define a variable ind to store the indexes needed to order the
 population values
[redacted]

# Create a data frame my_df with the state name and its rank and
 ordered from least populous to most
[redacted]


# # NA
```

```
# Using new dataset
[redacted]

# Checking the structure
[redacted]

# Find out the mean of the entire dataset
[redacted]

# Use is.na to create a logical index ind that tells which entries
 are NA
[redacted]

# Determine how many NA ind has using the sum function
[redacted]


# # REMOVING NAs

# Note what we can do with the ! operator
[redacted]

# Create the ind vector
[redacted]

# We saw that this gives an NA
[redacted]

# Compute the average, for entries of na_example that are not NA
[redacted]




# # # ASSESSMENT 2.3: VECTOR ARITHMETIC

# # VECTORIZED OPERATIONS

# Assign city names to `city`
[redacted]
```

```
[redacted]

# Store temperature values in `temp`
[redacted]

# Convert temperature into Celsius and overwrite the original values
 of 'temp' with these Celsius values
[redacted]
# Create a data frame `city_temps`
[redacted]

# Define an object `x` with the numbers 1 through 100
[redacted]

# Load the data
[redacted]

# Store the per 100,000 murder rate for each state in murder_rate
[redacted]
# Calculate the average murder rate in the US
[redacted]



# # # SECTION 2 ASSESSMENT

# # Q1
[redacted]

# # Q2
[redacted]

# # Q3
[redacted]

[redacted]
```

[redacted]