Programlama Dilleri Laboratuvar Föyü
Ruby Programlama Dili 2
String, Dizi ve Hash Türleri

Şevket Umut ÇAKIR

1 Giriş

Bu deneyde Ruby programlama dilindeki metinler(string), diziler(array) ve hash yapısı incelenecektir.

2 Konu Anlatımı ve Deney Hazırlığı

2.1 Metinler(String)

Ruby programlama dilinde metinler çift veya tek tırnak sembolü ile tanımlanır. Tek tırnaklı metinler içinde çift tırnak sembolü ya da tam tersi kaçış sembolüne ihtiyaç duymadan kullanılabilir. Çift tırnak, daha genel bir metin tanımlama sembolüdür ve tek tırnağa göre daha fazla biçimde kullanılır. Metinleri biçimlendirmek için % sembolü, ara değerlemek için de #{} sembolleri kullanılır. Aşağıda örnek kullanımları vardır.

```
puts "Türkiye'nin başkenti Ankara'dır."
puts 'Bana "merhaba" dedi!'
puts "Bugün %s ayının %d. haftasının %d. günü" % ["Mart", 3, 3]
x=3.14
puts "x değişkeninin değeri: #{x}, iki katı: #{2*x}"
```

2.1.1 String Metotları

- `size`, `length`: Metnin uzunluğunu verir
- `upcase` ve `donwcase`: Metni büyük ve küçük harfe dönüştürür
- `strip`: Metnin başındaki ve sonundaki boşlukları siler
- `start_with?` ve `end_with?`: Metnin bir başka metinle başladığını ve bittiğini sorgular
- `split`: Metni parçalara böler ve bir dizi döndürür

- `join`: Dizi elemanlarını birleştirerek bir metin oluşturur
- `to_i`, `to_f` ve `to_s`: String, ondalıklı sayı ve string türleri arasında dönüşüm gerçekleştirir
- `<<`: Metnin sonuna başka bir metin ekler
- `each_char`: Karakterler üzerinden döngü oluşturur
- `gsub`: Metnin içindeki bir kısmı başka bir metin ile değiştirir
- `chomp`: Son karakteri siler

2.2 Diziler

Diziler(arrays), herhengi türden nesnelerin ardışık, tamsayı indeksli bir koleksiyonudur. Dizi indeksleri C ve Java'da olduğu gibi 0'dan başlar, Python dilindeki gibi elemanlara ters sırada erişmek için neagtif indeks kullanılabilir. Ruby dizileri, diğer dillerdeki gibi değişemez(boyutu) değildir, eleman eklenip silinebilir. Aşağıda dizi oluşturma örnekleri verilmiştir:

```
names = Array.new # Boş dizi(sıfır elemanlı)
names = Array.new(20) # 20 elemanlı boş(nil) değerli dizi
names = Array.new(4, "mac") # ["mac", "mac", "mac", "mac"]
nums = Array.new(10) { |e| e = e * 2 } # [0, 2, 4, 6, 8, 10, 12,
↪  14, 16, 18]
nums = Array.[](1, 2, 3, 4, 5) # [1, 2, 3, 4, 5]
nums = Array[1, 2, 3, 4, 5] # [1, 2, 3, 4, 5]
digits = Array(0..9) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

2.2.1 Dizi Operatörleri

- `*`: İkinci parametre tamsayı ise tekrarlar, string ise birleştirme yapar(join)
- `+`: Dizileri birleştirir
- `-`: İlk dizideki elemanları ikinci dizidekileri çıkartıp verir
- `&`: Kesişim, iki dizideki ortak elemanları, tekrar olmadan, döndürür
- `|`: Birleşim işlemi, tekrarlı değer içermez
- `<<` Dizinin sonuna eleman ekler, diziyi değiştirir
- `==` İki dizi aynı sayıda eleman içeriyorsa ve elemanlar birbirine eşitse `true` döndürür
- `[]`: Dizinin elemanlarına erişim

```ruby
[1,2,3] & [3,4,5] # [3]
[1,2,3] | [3,4,5] # [1, 2, 3, 4, 5]
[1,2,3] + [3,4,5] # [1, 2, 3, 3, 4, 5]
[1,2]*3 # [1, 2, 1, 2, 1, 2]
[1,2,3,4]*" -- " # "1 -- 2 -- 3 -- 4"
dizi = Array(1..20) # 20 dahil
dizi[3] # 4
dizi[-2] # 19
dizi[3..5] # [4, 5, 6]
dizi[3, 5] # [4, 5, 6, 7, 8]
```

2.2.2 Dizi Metotları

- `at(index)`: Verilen indisteki elemanı döndürür
- `delete(obj)`, `delete_if {|item| block}`: Eleman siler
- `each { |item| block }`, `each_index { |index| block }`: Elemanlar için each döngüsü
- `each_with_index {|item,index| block}`: İndeksli each döngüsü
- `index(obj)`: Elemanın indeksini verir, bulunamazsa `nil` verir
- `join(sep)`: Elemanları ayraç kullanarak bir metin olarak birleştirir
- `length`, `size`: Eleman sayısını verir
- `push`, `pop`, `shift` ve `unshift`: Sona ve başa eleman ekler ve siler
- `reverse`: Elemanların sırasını tersine çevirir
- `sort`: Diziyi sıralar

```ruby
dizi=Array(1..5)
dizi.delete(4) # [1, 2, 3, 5]
Array(1..20).delete_if {|item| item%2==0} # [1, 3, 5, 7, 9, 11,
↪  13, 15, 17, 19]
[1,2,3].each {|item| puts item*2} # Alt alta 2 4 6 yazar
Array(1..5).reverse.each_with_index do |item,index|
    puts "#{index}:#{item}"
end
=begin
0:5
1:4
2:3
3:2
4:1
=end
```

2.3 Hash

Anahtar ve değer çiftlerinden oluşan verileri saklamak için kullanılan bir veri yapısıdır. Python dilindeki sözlük(dictionary) veri türüne benzer.

```ruby
# Hash oluşturma
h = Hash["a" => 100, "b" => 200]
# Anahtar olarak sembol kullanma
h2 = Hash.new
h2[:a] = 100
h2[:b] = 200
puts h["a"] # 100
puts h2[:b] # 200
```

2.3.1 Hash Metotları

```ruby
h = Hash["a" => 100, "b" => 200]
h.has_key? "a" # true
h.has_value? 300 # false
h.each {|key, value| puts "#{key}:#{value}"}
h.keys # ["a", "b"]
h.values # [100, 200]
```

3 Deneyin Uygulanması

Bu kısımda deneyde gerçeklenmesi gereken kodlar anlatılmaktadır.

3.1 Sabit Başlangıç

Kullanıcıdan alınan metnin ilk harfi sabit kalmak koşulu ile içindeki ilk harfle aynı olan harfleri "*" sembolü ile değiştiren kodu yazınız. Örneğin *google* girdisi *goo*le* olarak çıktı verilmelidir.

3.2 Karıştır

Kullanıcıdan alınan iki adet metnin ilk iki harflerini yer değiştirerek birleştiren ve ekrana yazan kodu yazınız. Örneğin **mixer** ve **glider** girdileri için çıktı **glxer miider** olmalıdır.

3.3 Önce x Sırala

Kullanıcıdan alınan satırları(boş satır gelene kadar) önce x içerenleri sonra diğerlerini sıralayarak ekrana yazdıran programın yazılması istenmektedir. Girdi metninin bittiğini belirtmek için son satır boş metin olarak gönderilmektedir. Aşağıda örnek girdi ve çıktı bulunmaktadır.

Girdi:	**Çıktı**:
`mix`	`xanadu`
`xyz`	`xyz`
`apple`	`aardvark`
`xanadu`	`apple`
`aardvark`	`mix`

3.4 En Çok Tekrarlayan Kelime

Kullanıcıdan alınan metinde en çok tekrar eden 5 kelime ve tekrar etme sayısının ekrana yazılması istenmektedir. Girdi metni işlenmeden önce nokta(.), virgül(,), kesme işareti(') boş string ile; tire işareti(-) ise bir adet boşluk karakteri ile yer değiştirmelidir. Ayrıca metinler küçük harfe dönüştürülüp işlenmelidir. Bu problemin çözümü için hash yapısı kullanılabilir. Sözlüğün anahtar değeri sözcüğü, değeri ise tekrar sayısını saklayacaktır.

Kaynaklar

[1] *Ruby Arrays.* URL: `https://www.tutorialspoint.com/ruby/ruby_arrays.htm` (son erişim: 19.3.2019).

[2] *Ruby Hashes.* URL: `http://rubylearning.com/satishtalim/ruby_hashes.html` (son erişim: 19.3.2019).

[3] *Ruby String Methods (Ultimate Guide).* URL: `https://www.rubyguides.com/2018/01/ruby-string-methods/` (son erişim: 19.3.2019).