

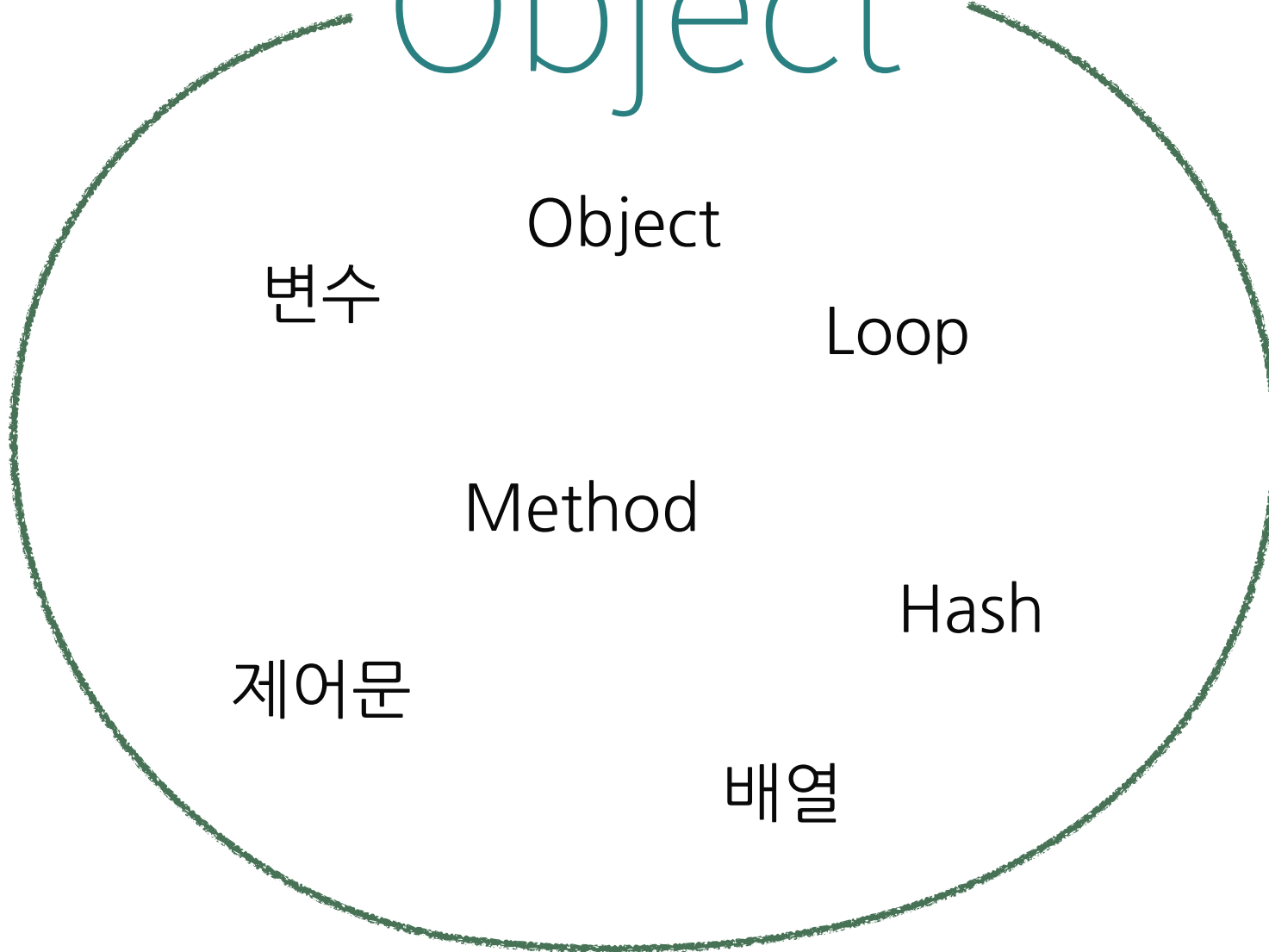
Ruby

2주차

수요일

멋쟁이사자처럼  건국대학교

Object



example.rb

#이거는한줄주석!

1. 한줄 주석

=begin

이렇게 하면 여러줄 주석을 할수 있어요!

2. 여러줄 주석

=end

example.rb

#이거는한줄주석!

1. 한줄 주석

=begin

이렇게 하면 여러줄 주석을 할수 있어요!

2. 여러줄 주석

=end

연산하기 - 1. 수연산 3/

```
$ irb
irb(main):001:0> 3**4
=> 81
irb(main):002:0> 4**2
=> 16
irb(main):003:0> 2**5
=> 32
irb(main):004:0> 5 % 3
=> 2
irb(main):005:0> 7 % 2
=> 1
```

← 1. 거듭제곱 구하기

← 2. 나눗셈 나머지 구하기

연산하기 - 1. 수연산 3/

```
$ irb
irb(main):001:0> 3**4
=> 81
irb(main):002:0> 4**2
=> 16
irb(main):003:0> 2**5
=> 32
irb(main):004:0> 5 % 3
=> 2
irb(main):005:0> 7 % 2
=> 1
```

← 1. 거듭제곱 구하기

← 2. 나눗셈 나머지 구하기

연산하기 - 1. 수연산 3/

```
$ irb
irb(main):001:0> var1 = 1
=> 1
irb(main):002:0> var1 = var1 + 4
=> 5
irb(main):003:0> var2 = 2
=> 2
irb(main):004:0> var2 += 4
=> 6
```

같은 연산방법

모든 수 연산자에 쓸수있다!
(무려!)

```
$ irb
irb(main):001:0> 3 == 4
=> false
irb(main):002:0> 4 >= 2
=> true
irb(main):003:0> 2 => 5
=> error
irb(main):004:0> 5 <= 3
=> false
irb(main):005:0> 7 =< 2
=> error
```

← 결과값이 boolean!

← '=' 기호는 나중에!


```
$ irb
irb(main):001:0> true && true
=> true
irb(main):002:0> true && false
=> false
irb(main):003:0> false && false
=> false
irb(main):004:0> true || true
=> true
irb(main):005:0> true || false
=> true
irb(main):006:0> false || false
=> false
```

← 하나만 true여도 True

← 하나만 false여도 False

```
$ irb
irb(main):001:0> (3 > 2) && (4 == 2*2)
=> true
irb(main):002:0> (9 < 3) && (5 > 2)
=> false
irb(main):003:0> (1 == 0) || (3 == 5)
=> false
irb(main):004:0> (3**2 > 4) || (5 == 2)
=> true
```

비교연산자 결과를
논리값으로 사용해서

논리연산에 사용!

example.rb

var1 = 1

var2 = 2

var3 = 3

def plus_four(num)

num += 4 # num = num + 4

return num

end

puts plus_four(var1) # 5

puts plus_four(var2) # 6

puts plus_four(var3) # 7

← 1. 변수 선언

example.rb

var1 = 1

var2 = 2

var3 = 3

def plus_four(num)

num += 4 # num = num + 4

return num

end

puts plus_four(var1) # 5

puts plus_four(var2) # 6

puts plus_four(var3) # 7

1. 변수 선언

2. 메소드 선언

example.rb

var1 = 1

var2 = 2

var3 = 3

def plus_four(num)

num += 4 # num = num + 4

return num

end

puts plus_four(var1) # 5

puts plus_four(var2) # 6

puts plus_four(var3) # 7

1. 변수 선언

2. 메소드 선언

3. 반환값

example.rb

var1 = 1

var2 = 2

var3 = 3

def plus_four(num)

num += 4 # num = num + 4

return num

end

puts plus_four(var1) # 5

puts plus_four(var2) # 6

puts plus_four(var3) # 7

1. 변수 선언

2. 메소드 선언

3. 반환값

4. 메소드 호출

자주 사용하는 메소드는
pdf를 참조하세요! (짱긋✨)

example.rb

```
puts "Hello Lions!"  
puts "Hello Lions!"  
puts "Hello Lions!"  
puts "Hello Lions!"  
puts "Hello Lions!"  
puts "Hello Lions!"  
puts "Hello Lions!"  
puts "Hello Lions!"  
puts "Hello Lions!"  
puts "Hello Lions!"
```

10번 타이핑
실화입니까..?



example.rb

‘조건에 쓰일 변수선언’

while 조건

실행할 코드

조건에 맞추기 위한 코드

end

← 1. true 일 동안 실행

example.rb

‘조건에 쓰일 변수선언’

while 조건

실행할 코드

조건에 맞추기 위한 코드

end

← 1. true 일 동안 실행

← 2. false로 만들어줄 코드

```
example.rb
```

```
for num in 범위  
  실행할코드  
end
```

← 코드실행횟수를 고려한 범위설정!

```
example.rb
```

```
for num in 1..3  
  puts num  
end
```

```
# => 1  
      2  
      3
```

← 1부터 3까지!

```
example.rb
```

```
num = 0
```

```
while num < 5
```

```
  if 조건
```

```
    puts num
```

```
  elsif 조건
```

```
    실행할 코드
```

```
  else
```

```
    실행할 코드
```

```
    num += 1
```

```
end
```

조건이 true 일경우만 실행!

1. 첫번째 조건

example.rb

```
num = 0
```

```
while num < 5
```

```
  if 조건
```

```
    puts num
```

```
  elsif 조건
```

```
    실행할 코드
```

```
  else
```

```
    실행할 코드
```

```
    num += 1
```

```
end
```

조건이 true 일경우만 실행!

1. 첫번째 조건

2. 조건이 두가지 이상 필요할 경우
if의 다음조건

example.rb

```
num = 0
```

```
while num < 5
```

```
  if 조건
```

```
    puts num
```

```
  elsif 조건
```

```
    실행할 코드
```

```
  else
```

```
    실행할 코드
```

```
    num += 1
```

```
  end
```

조건이 true 일경우만 실행!

1. 첫번째 조건

2. 조건이 두가지 이상 필요할 경우
if의 다음조건

3. 모든 조건이 맞지 않을 경우
실행할 코드

```
num = 0 ('조건에 사용될 변수')
```

```
while num < 5
```

```
  case num
```

```
  when 1
```

```
    실행할코드
```

```
  when 2
```

```
    실행할코드
```

```
  when 3
```

```
    실행할코드
```

```
  when 4
```

```
    실행할코드
```

```
  else
```

```
    puts num
```

```
  num += 1
```

```
end
```

case의 변수가 when의 값과
같을때!

1. num을 기준으로

조건문 - case문 11/

```
num = 0 ('조건에 사용될 변수')
```

```
while num < 5
```

```
  case num
```

```
  when 1
```

```
    실행할코드
```

```
  when 2
```

```
    실행할코드
```

```
  when 3
```

```
    실행할코드
```

```
  when 4
```

```
    실행할코드
```

```
  else
```

```
    puts num
```

```
  num += 1
```

```
end
```

case의 변수가 when의 값과
같을때!

1. num을 기준으로

2. num == 1일때

```
num = 0 ('조건에 사용될 변수')
```

```
while num < 5
```

```
  case num
```

```
  when 1
```

```
    실행할코드
```

```
  when 2
```

```
    실행할코드
```

```
  when 3
```

```
    실행할코드
```

```
  when 4
```

```
    실행할코드
```

```
  else
```

```
    puts num
```

```
  num += 1
```

```
end
```

case의 변수가 when의 값과
같을때!

1. num을 기준으로

2. num == 1일때

3. num == 2일때

```
num = 0 ('조건에 사용될 변수')
```

```
while num < 5
```

```
  case num
```

```
  when 1
```

```
    실행할코드
```

```
  when 2
```

```
    실행할코드
```

```
  when 3
```

```
    실행할코드
```

```
  when 4
```

```
    실행할코드
```

```
  else
```

```
    puts num
```

```
  num += 1
```

```
end
```

case의 변수가 when의 값과
같을때!

1. num을 기준으로

2. num == 1일때

3. num == 2일때

4. num == 3일때

```
num = 0 ('조건에 사용될 변수')
```

```
while num < 5
```

```
  case num
```

```
  when 1
```

```
    실행할코드
```

```
  when 2
```

```
    실행할코드
```

```
  when 3
```

```
    실행할코드
```

```
  when 4
```

```
    실행할코드
```

```
  else
```

```
    puts num
```

```
  num += 1
```

```
end
```

case의 변수가 when의 값과
같을때!

1. num을 기준으로

2. num == 1일때

3. num == 2일때

4. num == 3일때

5. num == 4일때

```
num = 0 ('조건에 사용될 변수')
```

```
while num < 5
```

```
  case num
```

```
  when 1
```

```
    실행할코드
```

```
  when 2
```

```
    실행할코드
```

```
  when 3
```

```
    실행할코드
```

```
  when 4
```

```
    실행할코드
```

```
  else
```

```
    puts num
```

```
  num += 1
```

```
end
```

case의 변수가 when의 값과
같을때!

1. num을 기준으로

2. num == 1일때

3. num == 2일때

4. num == 3일때

5. num == 4일때

6. num == 5일때

데이터를 저장된 순서로 불러올 수 있다!

건대멋사 []

데이터를 저장된 순서로 불러올 수 있다!




건대멋사 [ 0]

데이터를 저장된 순서로 불러올 수 있다!

건대멋사 [ ]





0 1

데이터를 저장된 순서로 불러올 수 있다!

건대멋사 [  ]

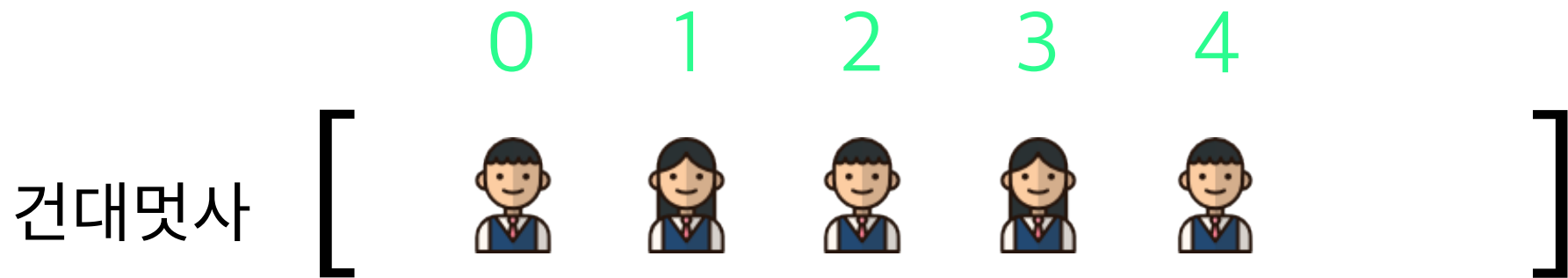
0 1 2

데이터를 저장된 순서로 불러올 수 있다!

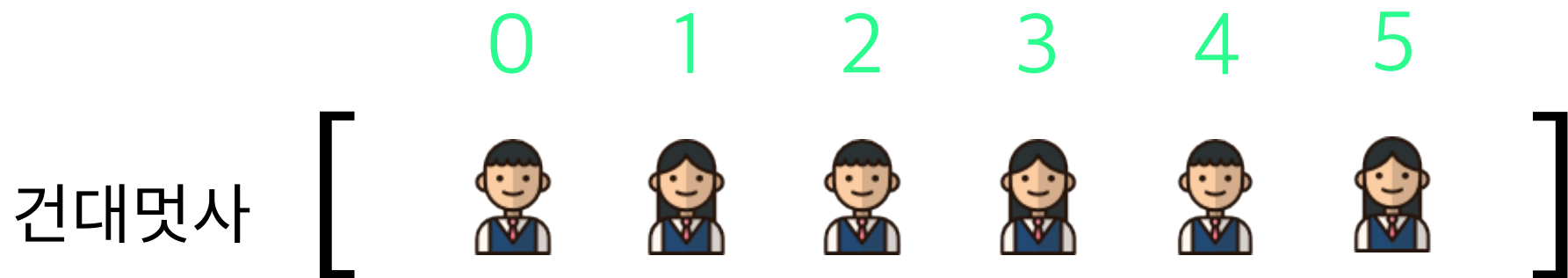
건대멋사 [   ]

0 1 2 3

데이터를 저장된 순서로 불러올 수 있다!



데이터를 저장된 순서로 불러올 수 있다!



데이터를 저장된 키로 불러올 수 있다!

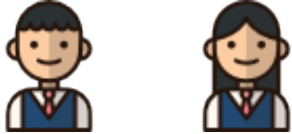
건대멋사 []

데이터를 저장된 키로 불러올 수 있다!

건대멋사 [ ruby]

데이터를 저장된 키로 불러올 수 있다!

건대멋사 [ruby js]



데이터를 저장된 키로 불러올 수 있다!

ruby js html

건대멋사

[



]

데이터를 저장된 키로 불러올 수 있다!

ruby js html css

건대멋사

[



]

데이터를 저장된 키로 불러올 수 있다!

ruby js html css rails

건대멋사







[



]

데이터를 저장된 키로 불러올 수 있다!

ruby js html css rails 멍멍

건대멋사 [     ]

example.rb

```
my_array = [1, "konkuk", true]
```

```
puts my_array[0] #=> 0
```

```
puts my_array[1] #=> 1
```

```
puts my_array[2] #=> 2
```

← 1. 배열 선언하기

example.rb

```
my_array = [1, "konkuk", true]
```

1. 배열 선언하기

```
puts my_array[0] #=> 0
```

2. index = 0 출력

```
puts my_array[1] #=> 1
```

```
puts my_array[2] #=> 2
```

index는 0부터 (갯수-1)까지!

example.rb

```
my_array = [1, "konkuk", true]
```

1. 배열 선언하기

```
puts my_array[0] #=> 0
```

2. index = 0 출력

```
puts my_array[1] #=> 1
```

3. index = 1 출력

```
puts my_array[2] #=> 2
```

index는 0부터 (갯수-1)까지!

example.rb

```
my_array = [1, "konkuk", true]
```

1. 배열 선언하기

```
puts my_array[0] #=> 0
```

2. index = 0 출력

```
puts my_array[1] #=> 1
```

3. index = 1 출력

```
puts my_array[2] #=> 2
```

4. index = 2 출력

index는 0부터 (갯수-1)까지!

배열에 요소추가하기는
pdf를 참조하세요! (짱긔💡💡)

example.rb

```
my_hash = {  
  key1: value1,  
  key2: value2,  
  key3: value3  
}
```

해시만들기 1

1. 해시 선언하기

example.rb

```
my_hash = {  
  key1: value1,  
  key2: value2,  
  key3: value3  
}
```

해시만들기 1

← 1. 해시 선언하기

← 2. 키-값쌍 넣어주기

띄어쓰기와 쉼표에 주의!

example.rb

```
my_hash = Hash.new
```

```
my_hash["key1"] = value1
```

```
my_hash["key2"] = value2
```

```
my_hash["key3"] = value3
```

해시만들기 2

1. 해시 선언하기

example.rb

```
my_hash = Hash.new
```

```
my_hash["key1"] = value1
```

```
my_hash["key2"] = value2
```

```
my_hash["key3"] = value3
```

해시만들기 2

1. 해시 선언하기

2. 키 지정과 동시에
값 넣어주기

example.rb

```
:my_sym.to_s
```

```
#=> "my_sym"
```

```
"likelion".to_sym
```

```
#=> :likelion
```

1. 해시에 to_s 메소드 사용

example.rb

`:my_sym.to_s`

`#=> "my_sym"`

`"likelion".to_sym`

`#=> :likelion`

1. 해시에 to_s 메소드 사용

2. 문자열에 to_sym 메소드 사용

example.rb

```
my_arr = [1, 2, 3, 4, 5]
```



1. 배열 선언

```
my_arr.each do |num|  
  num *= 2  
  puts num  
end
```

example.rb

```
my_arr = [1, 2, 3, 4, 5]
```

1. 배열 선언

```
my_arr.each do |num|
```

2. each 메소드 적용

```
  num *= 2
```

```
  puts num
```

```
end
```


example.rb

```
my_arr = [1, 2, 3, 4, 5]
```

1. 배열 선언

```
my_arr.each do |num|
```

2. each 메소드 적용

```
  num *= 2
```

```
  puts num
```

3. 각각 수의 2배를 출력

```
end
```

example.rb

```
my_hash = {  
  key1: value1,  
  key2: value2,  
  key3: value3  
}
```

```
my_hash.each do |x, y|  
  puts key  
  print value  
end
```

← 1. 해시 선언

example.rb

```
my_hash = {  
  key1: value1,  
  key2: value2,  
  key3: value3  
}
```

← 1. 해시 선언

```
my_hash.each do |x, y|  
  puts key  
  print value  
end
```

← 2. each 메소드 적용

example.rb

```
my_hash = {  
  key1: value1,  
  key2: value2,  
  key3: value3  
}
```

1. 해시 선언

```
my_hash.each do |x, y|  
  puts key  
  print value  
end
```

2. each 메소드 적용

3. 해시에서는 each 사용시
key, value 순으로 찾도록
고정되어있다.

✨꼭!✨

✨pdf에 있는 메소드들 봐야해요✨

✨배열과 해시 다루는 부분도!✨

멋쟁이사자처럼



건국대학교