Lesson 2 Ruby tasks

Task 1: Метод со смешанной сигнатурой

Description:

Реализовать метод **mega_sum**, который первым аргументом принимает массив и реализует следующее поведение:

- mega_sum([1, 2, 3, 4]) # => 10 # простая сумма элементов, для пустого массива должна быть нулевой
- mega_sum([1, 2, 3, 4], 10) # => 20 # сумма всёх элементов + переданное значение
- mega_sum([1, 2, 3, 4]) { |i| i ** 2 } => 30 # сумма элементов с применённым блоком
- mega_sum([1, 2, 3, 4], 10) { |i| i ** 2 } => 40 # сумма элементов с применённым блоком + переданное значение

Это всё один метод!

Task 2: Monkey patching

Description:

Апгрейд стандартной библиотеки. Добавить метод массивам, который будет возвращать все чётные числа, соответсвующие переданному в блок условию:

Чтобы это работало, делаем так:

```
class Array
  def even_search
     < your code is here >
  end
end
```

Пример:

```
[1, 2, 3, 4, 5, 6, 7].even_search { |i| i > 2 } # => [4, 6]
[1, 2, 3, 4, 5, 6, 7].even_search { |i| i > 10 } # => []
[2, 4, 6, 8, 10, 12, 7].even_search { |i| i.between?(6, 12) } # => [6, 8, 10, 12]
```

Task 3: Поиск в коллекциях

Description:

Реализовать метод **`range_finder`** который получая заданый диапазон возвращает максимальное число у которого остаток деления на 7 равен 3.

При его отсутсвии такого числа возвращает nil.

Пример:

```ruby
def range_finder(range)
  < your code is here >
end

range_finder((1..10)) # => 10
range_finder((50...105)) # => 101
range_finder((1..5)) # => 3
```

Task 4: Поиск подстрок

Description:

Написать метод **`ip_scapper`** который принимает строку, и список ip-адресов, а возвращает массив всех найденых в строке из этого списка.

Пример:

```ruby
ip_scapper(
 'root: 111.25.1.1\nlocal: 10.1.1.10',
 ['111.25.1.1', '10.1.1.10']
) # => ['111.25.1.1', '10.1.1.10']

ip_scapper(
 'root: 111.25.1.1\nlocal: 10.1.1.10',
 ['1.1.1.1']
) # => []

ip_scapper(
 'some fake 999.666.10.1 and real 233.100.1.100',
 ['233.100.1.100']
) # => ['233.100.1.100']
```

Task 5: Санитайзер

Description:

Обрабатывая поток логов на нужно прятать все наши пароли.

Реализовать метод sanitize_password!, который получает на вход строку, и изменяет её таким образом, что всё что находится после ключевого слова: password: и до следуюещего слова должно быть заменено на ****** (6 звёздочек).

Пример:

```
s = "{server:\nuser:root\npassword:!@$#@\nhost:localhost"
sanitize_password!(s)
p s # => "{server:\nuser:root\npassword:******\nhost:localhost"

s = '
  2017-04-14 09:14:02.5 ubuntu-yakkety[13310] password: "hello"
  Warning - Agent failed to register with core!
  password:admin
'
sanitize_password!(s)
p s # => "\n  2017-04-14 09:14:02.5 ubuntu-yakkety[13310] password:******\n
Warning - Agent failed to register with core!\n  password:******\n
```

Task 6: Processing chain

Description:

Реализовать метод **build** принимающий аргумент в виде строки и блок. Он должен пробрасывать начальное значение вместе с блоком методу **a**, который в свою очередь, также принимая аргумент и блок, выполняет этот блок с полученным значением и собственным аргументом равным имени метода в виде строки - "a". После процессинга метод **a** прокидывает по-аналогии результат и блок методу **b**, который работает также, но уже со значением "b" и передает вычисления методу **c**, который, отработав, уже просто выводит итоговый результат.

Аналогия из жизни: процессинг ассетов для веба, проходящий через стадии конкатенации, минификации и тд.

Пример работы:

```
initial = ""
result = build(initial) { |string, value| string + value }
p result # => "abc"
# пустая строка получает "a" от метода :a, "b" от метода :b и "c" от метода :c

# при этом, к примеру:
c("cianci") { |string, value| string.count(value) }
# => 2 (так как в строке 2 символа "c")
```