Homework from Class 2

Homework - Strings

- Given a string, return true if it is a palindrome, false otherwise.
  - Remove blank spaces and punctuation using tr.
  - Use the ternary operator. (condition ? true_value : false_value)

One solution:

```
str = "Able was I ere I saw Elba"

str1 = str.tr(" ", "").downcase
str2 = str1.reverse

str1 == str2 ? print(str + ": this is a palindrome.") : print(str + ": this is not a palindrome.”)
```

Another solution:

```
def is_palindrome?(arg)
   # strip string of non-word characters, force lowercase
   process_string = arg.gsub(/[^\w]/, '').downcase
   # check whether it's the same forward and backwards
   return process_string === process_string.reverse
end
```

[^\w] => \w stands for “word character”. The ^ in this case negates \w and finds all non-word characters.   gsub then replaces them with the empty string.
An alternate way of doing this is to use [\W] - which is short for [^\w], the negated version of \w.
What to search for on Google: regexp \w

Another solution: this one works, but doesn’t remove punctuation:

```
def palindrome?(sentence)
  sentence_downcase_nopunct = sentence.downcase.gsub(" ", "")
  if sentence_downcase_nopunct  ==  sentence_downcase_nopunct.reverse
    true
  else
   false
  end
end
```

sentence.downcase.gsub(/[,'"?]/, “") will remove punctuation marks as well.

Homework - Arrays

Given an array of strings representing the first names of people in a class. Return an array with two elements: the first element is a new array with class members whose names begin a letter between “A” and “J”. The second element is a new array with those whose names begin with a letter between “K” and “Z” inclusive.

- Example
  a=%w{ Annette Laura Louise Elizabeth Jessica Cheryl Janice Rebecca Kay Shannon AnnMarie Hailey Stephanie }
- Result: [["Annette", "Elizabeth", "Jessica", "Cheryl", "Janice", "AnnMarie", "Hailey"], ["Laura", "Louise", "Rebecca", "Kay", "Shannon", “Stephanie"]]

One solution:

```
def sort_names(a)
   # set up arrays to sort names into
   a_j = []
   k_z = []
   a.map{ |name| name.chars.first < "K" ? a_j.push(name) : k_z.push(name)}
   return [a_j, k_z]
end
```

Result:

```
irb(main):046:0> sort_names(%w{ Annette Laura Louise Elizabeth Jessica Cheryl
Janice Rebecca Kay Shannon AnnMarie Hailey Stephanie })
=> [["Annette", "Elizabeth", "Jessica", "Cheryl", "Janice", "AnnMarie",
"Hailey"], ["Laura", "Louise", "Rebecca", "Kay", "Shannon", “Stephanie"]]
```

Using the Enumerator#partition statement:

```
irb(main):088:0> a.partition{|name| name[0] < "K"}
=> [["Annette", "Elizabeth", "Jessica", "Cheryl", "Janice", "AnnMarie",
"Hailey"], ["Laura", "Louise", "Rebecca", "Kay", "Shannon", “Stephanie"]]
```

This one almost works:

```
array = %w{ Annette Laura Louise Elizabeth Jessica Cheryl Janice Rebecca Kay Shannon
AnnMarie Hailey Stephanie Celia }

a_j_array = []
k_z_array = []

a_j_array.push(array.select{ |a| a < "K" })
k_z_array.push(array.select{ |a| a > "J" })

p [a_j_array, k_z_array]
```

The problem is that Jessica and Janice show up in both arrays. One fix is:

```
a_j_array.push(array.select{ |a| a < "K" })
k_z_array.push(array.select{ |a| a > = "K" })
```

Another Solution:

```
names = ['Annette', 'Laura', 'Louise', 'Elizabeth', 'Jessica', 'Cheryl', 'Janice', 'Rebecca', 'Kay',
'Shannon', 'AnnMarie', 'Hailey', 'Stephanie']

first = names.select { |n| ('A'..'J').include?(n[0]) }
second = names.select { |n| ('K'..'Z').include?(n[0]) }

print [first, second]
```

We're using print on the last line, because puts takes the array [first,second] and calls the to_string method on it. We don't want the string, just the array.