JumpStart Live Day 4

Classroom exercises

Instructor code review

From day 3 exercises

You don't trust your users. Modify the program below to require the user to enter the same value twice in order to add that value to the total.

```
puts "Hello! We are going to total some numbers!"
puts "Enter a negative number to quit."

total = 0
input = gets.chomp.to_i
      input > -1
  total += input
  input = gets.chomp.to_i


puts "Result: #{total}"
```

Instructor code review

What could make this code better?

```
puts "Hello! We are going to total some numbers!"
puts "Each number should be entered twice to be accepted."
puts "Enter a negative number (twice) to quit."

total = 0
input = gets.chomp.to_i
confirmedInput = gets.chomp.to_i

      true # infinite loop, quit on break
    input == confirmedInput
      input < 0 # time to quit

            # add to total
          total += input

    # get new inputs
    input = gets.chomp.to_i
    confirmedInput = gets.chomp.to_i
      # inputs don't match, assume confirmedInput to be new input
    input = confirmedInput
    confirmedInput = gets.chomp.to_i



puts 'Result: #{total}'
```

What does the code below do?

```
print "Enter a number between 1 and 10: "
number = gets.chomp.to_i

      number < 1 || number > 10
    print "Not in range. Enter a number between 1 and 10: "


number = gets.chomp.to_i
puts "#{number} it is, then!"
```

What does the code below do?

```
print "Enter a number between 1 and 10: "
number = gets.chomp.to_i

      number < 1 || number > 10
    print "Not in range. Enter a number between 1 and 10: "


number = gets.chomp.to_i
puts "#{number} it is, then!"
```

```
print "Enter a number between 1 and 10: "
number = gets.chomp.to_i

      number < 1 || number > 10
    print "Not in range. Enter a number between 1 and 10: "
    number = gets.chomp.to_i


puts "#{number} it is, then!"
```

Convert to an `until` loop

```
print "Enter a number between 1 & 10: "
number = gets.chomp.to_i

      number < 1 || number > 10
    print "Not in range. Try again: "
    number = gets.chomp.to_i


puts "#{number} it is, then!"
```

Convert to an `until` loop

```
print "Enter a number between 1 & 10: "
number = gets.chomp.to_i

      number < 1 || number > 10
    print "Not in range. Try again: "
    number = gets.chomp.to_i


puts "#{number} it is, then!"
```

```
# Version 1
print "Enter a number between 1 & 10: "
number = gets.chomp.to_i

# simply negate the while condition
     !(number < 1 || number > 10)
    print "Not in range. Try again: "
    number = gets.chomp.to_i


puts "#{number} it is, then!"
```

Convert to an `until` loop

```
print "Enter a number between 1 & 10: "
number = gets.chomp.to_i

      number < 1 || number > 10
    print "Not in range. Try again: "
    number = gets.chomp.to_i


puts "#{number} it is, then!"
```

```
# Version 1
print "Enter a number between 1 & 10: "
number = gets.chomp.to_i

# simply negate the while condition
     !(number < 1 || number > 10)
    print "Not in range. Try again: "
    number = gets.chomp.to_i


puts "#{number} it is, then!"
```

```
# Version 2
print "Enter a number between 1 & 10: "
number = gets.chomp.to_i

     number >= 1 && number <= 10
    print "Not in range. Try again: "
    number = gets.chomp.to_i


puts "#{number} it is, then!"
```

`.times` loops and `.each` iterators

What will get printed?

```
my_array = ["apples", "bananas", "oranges"]

my_array.count.times    |x|
    puts x


my_array.each    |x|
    puts x
```

`.times` loops and `.each` iterators

What will get printed?

```
my_array = ["apples", "bananas", "oranges"]

my_array.count.times    |x|
    puts x


my_array.each    |x|
    puts x
```

```
brooks@ada:~$ ruby fruit.rb
0
1
2
apples
bananas
oranges
```

`.times` loops and `.each` iterators

What will get printed?

```
my_array = ["apples", "bananas", "oranges"]

my_array.count.times    |x|
    puts x


my_array.each    |x|
    puts x
```

```
brooks@ada:~$ ruby fruit.rb
0
1
2
apples
bananas
oranges
```

How can I use the `.times` loop to print the fruit names?

`.times` loops and `.each` iterators

What will get printed?

```
my_array = ["apples", "bananas", "oranges"]

my_array.count.times    |x|
    puts x


my_array.each    |x|
    puts x
```

```
brooks@ada:~$ ruby fruit.rb
0
1
2
apples
bananas
oranges
```

How can I use the `.times` loop to print the fruit names?

```
my_array.count.times    |x|
    puts my_array[x]
```

Choose descriptive variable names

```
my_array = ["apples", "bananas", "oranges"]

my_array.count.times    |x|
    puts x


my_array.each    |x|
    puts x
```

vs

```
fruit = ["apples", "bananas", "oranges"]

fruit.count.times    |i|
    puts i


fruit.each    |fruit|
    puts fruit
```

Choose descriptive variable names

```
my_array = ["apples", "bananas", "oranges"]

my_array.count.times    |x|
    puts x


my_array.each    |x|
    puts x
```

vs

```
fruit = ["apples", "bananas", "oranges"]

fruit.count.times    |i|
    puts i


fruit.each    |fruit|
    puts fruit
```

is very commonly used as the index variable when looping

Fence post problem

Complete the following program - create a fence post based on user input

```
puts "Let's create a fence. We need at least 2 fence posts"
print "How many fence posts would you like? "
num_posts = gets.chomp.to_i

# assume that the user enters a valid integer input > 2
```

Fence post problem

Complete the following program - create a fence post based on user input

```
puts "Let's create a fence. We need at least 2 fence posts"
print "How many fence posts would you like? "
num_posts = gets.chomp.to_i

# assume that the user enters a valid integer input > 2
```

```
How many fence posts would you like? 2
|--|
```

```
How many fence posts would you like? 10
|--|--|--|--|--|--|--|--|--|--|
```

Fence post problem

Complete the following program - create a fence post based on user input

```
puts "Let's create a fence. We need at least 2 fence posts"
print "How many fence posts would you like? "
num_posts = gets.chomp.to_i

# assume that the user enters a valid integer input > 2
```

```
How many fence posts would you like? 2
|--|
```

```
How many fence posts would you like? 10
|--|--|--|--|--|--|--|--|--|
```

```
# Solution 1
(num_posts.to_i - 1).times
    print "|--"


# add the final post "|" without a "-"
puts "|"
```

```
# Solution 2
print "|" # place starting post

(num_posts.to_i - 1).times
    print "--|"


puts # add a final newline
```

Exercise: fence post

Complete the following program

```
puts "I'm throwing a party for my friends!"
print "Let me tell you who's coming... "
friends = ["Harry", "Hermione", "Ron", "Luna"]

count = friends.count
```

List all the friends in a sentence. Their names should appear on the same line, and without any surrounding brackets. For example:

```
brooks@ada:~$ ruby harry.rb
I'm throwing a party for my friends!
Let me tell you who's coming... Harry, Hermione, Ron, Luna.
```

```
brooks@ada:~$ ruby harry.rb
I'm throwing a party for my friends!
Let me tell you who's coming... Harry, Hermione, Ron and Luna.
```

```
brooks@ada:~$ ruby harry.rb
I'm throwing a party for my friends!
Let me tell you who's coming... Harry, Hermione, Ron, and Luna.
```

Fence post - solution 1

```
puts "I'm throwing a party for my friends!"
print "Let me tell you who's coming... "
friends = ["Harry", "Hermione", "Ron", "Luna"]
```

```
# first friend name outside the loop
print friends[0]

# loop through the rest of the friends and print them out
(friends.count - 1).times    |i|
    print ", #{friends[i + 1]}"


puts "."
```

```
brooks@ada:~$ ruby harry.rb
I'm throwing a party for my friends!
Let me tell you who's coming... Harry, Hermione, Ron, Luna.
```

Fence post - solution 2

```
puts "I'm throwing a party for my friends!"
print "Let me tell you who's coming... "
friends = ["Harry", "Hermione", "Ron", "Luna"]
```

```
# first friend name outside the loop
print friends[0]

# loop through the all but two friends
(friends.count - 2).times    |i|
    print ", #{friends[i + 1]}"


# print the last friend
puts " and #{friends[-1]}"
```

```
brooks@ada:~$ ruby harry.rb
I'm throwing a party for my friends!
Let me tell you who's coming... Harry, Hermione, Ron and Luna.
```

Fence post - solution 3

```
puts "I'm throwing a party for my friends!"
print "Let me tell you who's coming... "
friends = ["Harry", "Hermione", "Ron", "Luna"]
```

```
# loop through all but one friend
(friends.count - 1).times    |i|
    print "#{friends[i + 1]}, "


# print the last friend
puts " and #{friends[-1]}"
```

```
brooks@ada:~$ ruby harry.rb
I'm throwing a party for my friends!
Let me tell you who's coming... Harry, Hermione, Ron, and Luna.
```

Boolean zen

Change the following pieces of code to follow [boolean zen]

```
# exercise 1
flag = true
puts "Flag is #{flag}"
     flag == true
      2 < 4 == true
        flag = false
        puts "Flag is now #{flag}"
```

```
# exercise 2
flag = false
puts "Flag is #{flag}"
     flag == false
      2 > 4 == false
        flag = true
        puts "Flag is now #{flag}"
```

Boolean zen

Change the following pieces of code to follow [boolean zen]

```
# exercise 1
flag = true
puts "Flag is #{flag}"
     flag == true
      2 < 4 == true
        flag = false
        puts "Flag is now #{flag}"
```

```
# exercise 2
flag = false
puts "Flag is #{flag}"
     flag == false
      2 > 4 == false
        flag = true
        puts "Flag is now #{flag}"
```

```
# solution 1
flag = true
puts "Flag is #{flag}"
     flag
      2 < 4
        flag = false
        puts "Flag is now #{flag}"
```

Boolean zen

Change the following pieces of code to follow [boolean zen]

```
# exercise 1
flag = true
puts "Flag is #{flag}"
     flag == true
      2 < 4 == true
        flag = false
        puts "Flag is now #{flag}"
```

```
# exercise 2
flag = false
puts "Flag is #{flag}"
     flag == false
      2 > 4 == false
        flag = true
        puts "Flag is now #{flag}"
```

```
# solution 1
flag = true
puts "Flag is #{flag}"
     flag
      2 < 4
        flag = false
        puts "Flag is now #{flag}"
```

```
# solution 2
flag = false
puts "Flag is #{flag}"
     !flag
      !(2 > 4) # or 2 <= 4
        flag = true
        puts "Flag is now #{flag}"
```

Get a jump start on day 5 exercises

https://github.com/Ada-Developers-Academy/jump-start-live/tree/master/lessons/day5#exercises

Debrief

- Why are arrays useful in programming?
- What topics are you still struggling with?
- What did you especially enjoy about today's class?
- What can be improved for future classes?