COMP 348
PRINCIPLES OF PROGRAMMING LANGUAGES

Tutorial #7

Object Programming with Ruby

RUBY PROGRAMMING

Tools:

https://www.jetbrains.com/ruby/download/#section=windows

https://www.ruby-lang.org/en/downloads/

Execute Ruby Online:

https://www.tutorialspoint.com/execute_ruby_online.php

https://repl.it/repls/HideousEffectiveComputeranimation

COMMAND TO COMPILE A PROGRAM

- Create a text file and save it with .rb extension.

  Open any editor and type the following.

  puts "Welcome to Ruby Programming"

  Save the file as: welcome_ruby.rb

- Command to compile the program:
  ruby welcome_ruby.rb

- Output:
  Welcome to Ruby Programming

INTERACTIVE RUBY

```
irb(main):001:0> 8+3+4
=> 15
irb(main):002:0> 9*9
=> 81
irb(main):003:0> x=5
=> 5
irb(main):004:0> y=x*4
=> 20
irb(main):005:0> z=y/2
=> 10
irb(main):006:0> x
=> 5
irb(main):007:0> 3+2*40/2-1
=> 42
irb(main):008:0> puts "Hello world"
Hello world
=> nil
irb(main):009:0>
```

puts will print out "Hello world", but irb will also print out the return value of **puts** - which is nil.

EXAMPLE OF IF CONSTRUCT

```
# program start
a = 10 * rand
if a < 5
puts "#{a} less than 5"
elsif a > 7
puts "#{a} greater than 7"
else
puts "#{a} Cheese sandwich!"
end
# program end
```

the **rand** function generates a random number between 0 and 1

EXAMPLE OF WHILE STATEMENT

```
$i = 1
$count = 6
while $i < $count  do
   puts("Inside the loop i = #$i" )
   $i +=2
```

<u>Output:</u>

Inside the loop i = 1
Inside the loop i = 3
Inside the loop i = 5

EXAMPLE OF CASE STATEMENT

```
a = (10*rand).round
#a = rand(11) would do the same
case a
when 0..5
puts "#{a}: Low"
when 6
puts "#{a}: Six"
else
puts "#{a}: Cheese toast!"
end
```

FUNCTION EXAMPLES

1/ Define a function called “welcome(name)” that will take name as input and output as “Hi <name>”

Creating a file name: “welcome_func.rb”

```
def welcome(name)
 puts "Hi #{name}"
end
welcome("visitor")
```

<u>Compile:</u>

```
ruby welcome_func.rb
```

<u>Output:</u>

Hi visitor

FUNCTION EXAMPLES

2/ Define a function called **multiply(a,b)** that produces product of 2 numbers.

```
def multiply(a,b)
 return product = a * b
end
puts multiply(6,7)
```

<u>Output:</u>

42

FUNCTION EXAMPLES

3/ Find the value of the following arguments given in the method “test”:

```
def test(a=1, b=2, c=a+2*b)
 puts "#{a}, #{b}, #{c}"
end

puts "calling test with no argument"
puts test
puts "\ncalling test with one argument"
puts test 9
puts "\ncalling test with two arguments"
puts test 2, 4
puts "\ncalling test with three arguments"
puts test 3, 7, 11
```

<u>Output:</u>

```
calling test with no argument
1, 2, 5

calling test with one argument
9, 2, 13

calling test with two arguments
2, 4, 10

calling test with three arguments
3, 7, 11
```

FUNCTION EXERCISE

1/ Define a function called “getCostAndMpg” will return at the same time: cost=30000 as AltimaCost & mpg=30 as AltimaMpg

?

<u>Output:</u>

?

ARRAY EXERCISES

Given the following array:

presidents = ["Ford", "Carter", "Reagan", "Bush", "Clinton"]

1/ Print the name of each element with a new line.

?

<u>Output:</u>

?

ARRAY EXERCISES (CONT..)

Given the following array:

presidents = ["Ford", "Carter", "Reagan", "Bush", "Clinton"]

2/ Delete last 2 elements of the array and print the remaining.

?

<u>Output:</u>

?

ARRAY EXERCISES (CONT..)

3/ Continue with Exercise2. Prepend the name of the following two presidents, Kennedy, and Johnson and print the array:

?

<u>Output:</u>

?

ARRAY EXERCISES (CONT..)

4/ Continue with Exercise3. Change the name of last president to “John Quincy Adams” and print the array:

?

Output:

?

ASSOCIATIVE ARRAY (HASH) EXERCISE

1/ Write a program using associative array to produce the following result set:

Lastname : Litt

Firstname : Steve

Social Security Number: 123456789

Corrected Social Security Number: 987654321

Gender : male

Hash length is 4

Hash class is Hash

ASSOCIATIVE ARRAY (HASH) EXERCISE

Solution:

?

DEFINING CLASSES AND FEATURES: NAMING

- Ruby uses a convention to help it distinguish the usage of a name: the first characters of a name indicate how the name is used.
- Class names, module names, and constants should start with an uppercase letter.
- Class variables start with two “at” signs (@@).
- Local variables, method parameters, and method names should all start with a lowercase letter or with an underscore( ).
- Global variables are prefixed with a dollar sign ($), while instance variables begin with a single “at" sign(@)

NAMING EXAMPLES

local_variable

CONSTANT_NAME / ConstantName / Constant_Name

:symbol_name

@instance_variable

@@class_variable

$global_variable

ClassName

method_name

ModuleName

OBJECTS

- Instances of classes (objects) contain state and behavior.
- Each object contains its own unique state.
- Behavior on the other hand is shared among objects.
- The state of the object is composed of a set of attributes (or fields), and their current values.

CLASS (EXERCISES)

Exercise#1

- Write a class **Box** which has 2 variables **width** and **height.**
- Define a constructor method to initialize their values.
- Then define two accessor methods **printWidth** and **printHeight** which will have the width and height respectively.
- Call these methods to print the width and height of the box by creating an instance of the class.

Solution

?

CLASS (EXERCISES)

Exercise#2

Now make 2 new setter methods **setWidth** for width and **setHeight** for height so that user can give their own values without using the initial values as before.

Solution

?