Explorations in Ruby: Practical
Practical 2

Student Name- Vipin Hans

Student Number- 17201230

1) In irb and each of the primitives ***class*** and ***instance_of?*** test the following to see what types of object they are and explain why you get the answers you do:

Answer-

a. “hello there big boy”

```
irb(main):003:0* "hello there big boy".class
=> String
irb(main):004:0>
```

The .class or .class.name returns the Class to which an object belongs.

```
irb(main):004:0> "hello there big boy".instance_of?(String)
=> true
irb(main):005:0>
```

if the object is an instance of that exact class, not a subclass.

b. 56

```
irb(main):006:0> 56.class
=> Integer
irb(main):007:0> 56.instance_of?(Integer)
=> true
irb(main):008:0> 56.instance_of?(String)
=> false
irb(main):009:0>
```

c. 34.00

```
irb(main):009:0> 34.00.class
=> Float
irb(main):010:0> 34.00.instance_of?(Float)
=> true
irb(main):011:0>
```

d. 0.222222354454365

```
irb(main):011:0> 0.222222354454365.class
=> Float
irb(main):012:0> 0.222222354454365.instance_of?(Float)
=> true
irb(main):013:0>
```

e. [“a”, “b”, “c”]

```
irb(main):013:0> ["a", "b", "c"].class
=> Array
irb(main):014:0> ["a", "b", "c"].instance_of?(Array)
=> true
```

```
irb(main):016:0> ["a", "b", "c"].instance_of?(String)
=> false
irb(main):017:0> ["a", "b", "c"].instance_of?(Integer)
=> false
```

f. +

```
+.class
  ^
        from C:/Ruby24-x64/bin/irb.cmd:19:in `<main>'
irb(main):019:0>
```

g. PI

```
irb(main):019:0> PI.class
NameError: uninitialized constant PI
        from (irb):19
        from C:/Ruby24-x64/bin/irb.cmd:19:in `<main>'
irb(main):020:0>
```

h. Math::PI

```
irb(main):020:0> Math::PI.class
=> Float
irb(main):021:0> Math::PI.instance_of?(Float)
=> true
irb(main):022:0>
```

i. add

```
irb(main):022:0> add.class
NameError: undefined local variable or method `add' for main:Object
        from (irb):22
        from C:/Ruby24-x64/bin/irb.cmd:19:in `<main>'
irb(main):023:0>
```

j. hellow

```
irb(main):023:0> hellow.class
NameError: undefined local variable or method `hellow' for main:Object
        from (irb):23
        from C:/Ruby24-x64/bin/irb.cmd:19:in `<main>'
irb(main):024:0>
```

k. hello = 8 and then check hello with ***class***

```
irb(main):024:0> hello=8
=> 8
irb(main):025:0> hello.class
=> Integer
irb(main):026:0> hello.instance_of?(Integer)
=> true
irb(main):027:0>
```

l. “goodbye”

```
irb(main):027:0> "goodbye".class
=> String
irb(main):028:0> "goodbye".instance_of?(String)
=> true
irb(main):029:0>
```

m. (56 + 45.32)

```
irb(main):029:0> (56 + 45.32).class
=> Float
irb(main):030:0> (56 + 45.32).instance_of?(Float)
=> true
irb(main):031:0>
```

n. (56 + 45)

```
irb(main):031:0> (56 + 45).class
=> Integer
irb(main):032:0> (56 + 45).instance_of?(Integer)
=> true
irb(main):033:0>
```

o. 5.to_s

```
irb(main):034:0> 5.to_s
=> "5"
irb(main):035:0> 5.to_s.class
=> String
irb(main):036:0> 5.to_s.instance_of?(String)
=> true
irb(main):037:0>
```

5.to_s converted this to String data type and then performing .class on it returned the new primitive data type class which is String. Instance_of?(String) returned Boolean value true or false.

p. “5”.to_i

```
irb(main):037:0> "5".to_i.class
=> Integer
irb(main):038:0> "5".to_i.instance_of?(Integer)
=> true
irb(main):039:0> "5".to_i.instance_of?(String)
=> false
irb(main):040:0>
```

q. five.to_s

```
irb(main):040:0> five.to_s
NameError: undefined local variable or method `five' for main:Object
        from (irb):40
        from C:/Ruby24-x64/bin/irb.cmd:19:in `<main>'
irb(main):041:0> five.to_s.class
NameError: undefined local variable or method `five' for main:Object
        from (irb):41
        from C:/Ruby24-x64/bin/irb.cmd:19:in `<main>'
irb(main):042:0>
```

2) In irb what happens when you evaluate the following. Try to predict it before trying them:

Answer-

a. “hello there big boy”.include?(“boy”)

Include? Returns true or false based upon if the part of string “boy” is included in String object “hello there big boy”. Therefore, in such case the value returned should be true.

```
irb(main):044:0> "hello there big boy".include?("boy")
=> true
irb(main):045:0>
```

b. “hello there big boy”.include(“ big”)

It should return some syntax error or method not defined since built-in method is include? And not include

```
irb(main):047:0> "hello there big boy".include(" big")
NoMethodError: undefined method `include' for "hello there big boy":String
Did you mean?  include?
        from (irb):47
        from C:/Ruby24-x64/bin/irb.cmd:19:in `<main>'
irb(main):048:0>
```

c. "hello there big boy".include?(" ere")
It will return **false** as there is a space " ere" in the string to be compared. If there were no space "ere" it should have been matched up with "there" in the String object "hello there big boy" and would have returned true in this scenario.

```
irb(main):051:0> "hello there big boy".include?(" ere")
=> false
irb(main):052:0> "hello there big boy".include?("ere")
=> true
irb(main):053:0>
```

d. What happens when you evaluate: ["a", "b", "c"] + ["d"]

```
irb(main):053:0> ["a", "b", "c"] + ["d"]
=> ["a", "b", "c", "d"]
irb(main):054:0>
```

Added new element to array. Array concatenation.

Array1 + Array2

Array1.concat Array2

e. What happens when you evaluate: ["a", "b", "c"] + "d"

It won't be added as the "d" object is of string type and not enclosed in [].

```
irb(main):055:0> ["a", "b", "c"] + "d"
TypeError: no implicit conversion of String into Array
        from (irb):55
        from C:/Ruby24-x64/bin/irb.cmd:19:in `<main>'
irb(main):056:0>
```

f. Is there an easy way to capitalise words, so "hello" becomes "Hello" ?

```
irb(main):060:0> "hello".capitalize
=> "Hello"
irb(main):061:0>
```

g. In the same vein, make “hello” “HELLO”.

```
irb(main):061:0> "hello".upcase
=> "HELLO"
irb(main):062:0>
```

h. Write a command to print out your name.

```
irb(main):063:0> puts "Vipin Hans"
Vipin Hans
=> nil
irb(main):064:0>
```

i. Write a method to print out your name.

```
//Method to print my name
def printname
        puts “Vipin Hans”
end
```

j. Write a method to print out any name.

```
def inputname
                        puts "Enter your name: "
                        name=gets
                        puts "Your name is: "+ name
end
```

k. Set up the varibles, *maxi*, *dick* and *twinko* so that they are all assigned numbers but two of them are assigned to the same numbers. Then show with a series of equality tests which ones actually have the same value.

```
def equalitytest
        maxi=10
        dick=10
```

```
        twinko=100

        if (maxi==dick)
            print "maxi and dick contains same value"
        elsif (dick==twinko)
            print "dick and twinko contains same value"
        elsif (twinko==maxi)
            print "twinko and maxi contains same value"
        else
            print "All values are different"
        end

end
```

l. If you change the variables with the same number to be a Float and Fixnum does it change the results of the equality tests ?

It doesn't really changes the result of program.

```
def equalitytest
        maxi=10
        dick=10.0
        twinko=100

        if (maxi==dick)
            print "maxi and dick contains same value"
        elsif (dick==twinko)
            print "dick and twinko contains same value"
        elsif (twinko==maxi)
            print "twinko and maxi contains same value"
        else
            print "All values are different"
        end
```

```
end
```

m. Do a version of these test using strings rather than numbers.

```
def equalitytest
maxi="maxi"
dick="maxi"
twinko="twinko"

if (maxi==dick)
        print "maxi and dick contains same value"
elsif (dick==twinko)
        print "dick and twinko contains same value"
elsif (twinko==maxi)
        print "twinko and maxi contains same value"
else
        print "All values are different"
end

end
```

3) What's a predicate ?

It returns boolean values which is either true or false such as include? And instance_of? In ruby such methods end with a question mark.

4) Define your own adding method that always adds 5 and 6 together.
So, my_add_five_to_six => 11.

```
def addnumbers

firstnumber=5
secondnumber=6
finalanswer=firstnumber+secondnumber
puts "Addition of 5 and 6 is: "+finalanswer.to_s
```

end

5) Put this defined method in a file and call it using the ruby command outside of Irb

- Save the file in any directory.
- Add Ruby installation bin path to environment variables.
- Open command prompt, and navigate to the respective directory where program is located.
- Execute command ruby filename.rb