1. Answer:

##1

## CONDITIONAL STATEMENTS

#if statement

ruby = "installed" # condition

if ruby == "installed"

puts "RubyMine is successfully installed. "

end

OUTPUT:

RubyMine is successfully installed.

#if else statement

gem = false # condition

print "Checking Gems for RubyMine ..."

puts

if gem == true

puts "Gems are avaliable."

else

puts "Gems need to be downloaded."

end

OUTPUT:

Checking Gems for RubyMine ...

Gems need to be downloaded.

#if else-if else statement

number = 12 # condition

if number > 0

puts "#{number} is a positive number."

elsif number < 0

puts "#{number} is a negative number."

else

puts "#{number} is a Zero."

end

OUTPUT:

12 is a positive number.

#if modifier

number = 12 # condition

puts "number is not 0" if number != 0

OUTPUTS:

number is not 0

#unless statement

number = 13 # condition

unless number/2 == 0

puts "#{number} is not an even number."

end

OUTPUT:

13 is not an even number.

#unless-else statement

number = 12 # condition

unless number/2 !=0

puts "#{number} is not an even number."

else

puts "#{number} is an even number."

end

OUTPUT:

12 is an even number.

#unless modifier

number = 12 # condition

puts "#{number} is not a negative number" unless number <0

OUTPUT:

12 is not a negative number

#case statement

number = 45

puts "Assuming number is a positive integer."

case number

when 0..9

puts "number is a one digit integer."

when 10..99

puts "number is a two digit integer."

else

puts "number is a three digit or more integer."

end

OUTPUT:

Assuming number is a positive integer.

number is a two digit integer.

#ternary operator

number = -20

str= number == 0?

"number is zero" : "number is not zero"

puts str

OUTPUT:

number is not zero

1. Answer:

##2

#LOOPS & ITERATORS

#WHILE-DO

greeting = ["Hi",","," How"," are"," you"," ?"]

index = 0

while index < 6 do

print greeting.at(index)

index +=1

end

puts

OUTPUT:

Hi, How are you ?

#WHILE MODIFIER

number = 5

print "#{number}"

print " #{number=number-1}" while number > 0

puts

OUTPUT:

5 4 3 2 1 0

#LOOP-DO

number = 1

loop do

if number > 10

break

end

print "#{number\*number} "

number += 1

end

puts

OUTPUT:

1 4 9 16 25 36 49 64 81 100

#UNTIL-DO

number = 2

print 25

until number == 11

print " #{number \* 25}"

number += 1

end

puts

OUTPUT:

25 50 75 100 125 150 175 200 225 250

#UNTIL MODIFIER

newyear = [" Year!"," New","Happy"]

print newyear.pop until newyear.empty?

puts

OUTPUT:

Happy New Year!

#FOR IN

hash = {1 => "fire", 2 => "water", 3 => "grass"}

for key,var in hash

puts "#{key} => #{var},"

end

puts

OUTPUT:

1 => fire,

2 => water,

3 => grass,

## ITERATORS ##

#DOWNTO

5.downto(1){|counter| print "#{counter} "}

puts

OUTPUT:

5 4 3 2 1

#UPTO

newyear = ["Happy ", "New ","Year ",",2017!!"]

0.upto(3){|n| print newyear.at(n)}

puts

OUTPUT:

Happy New Year ,2017!!

#TIMES

3.times{print "Bloody Mary... "}

puts

OUTPUT:

Bloody Mary... Bloody Mary... Bloody Mary...

#EACH

newyear= ["Good ","Bye ", 2016]

newyear.each{|component| print component}

puts

OUTPUT:

Good Bye 2016

#MAP

(1..10).map{|x| print "#{x\*x} "}

puts

OUTPUT:

1 4 9 16 25 36 49 64 81 100

#STEP

1.step(2,0.2) do |x| print "#{x} " end

puts

OUTPUT:

1. 1.2 1.4 1.6 1.8 2.0

#COLLECT

even = (1..5).collect{|b| b\*2}

print even

puts

OUTPUT:

[2, 4, 6, 8, 10]

#SELECT

number = (1..20).select{|b| b%7 == 0}

print number

puts

OUTPUT:

[7, 14]

#REJECT

number = (1..20).reject{|b| b%7 == 0}

print number

OUTPUT:

[1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20]

1. Answer:

## METHOD

def mean\_sigma(values)

sum1 = 0.0

for data in values do sum1 += data end

mean = sum1/values.length

sum2 = 0.0

for data in values do sum2 += (data - mean)\*\*2 end

stdev = Math::sqrt(sum2/values.length)

return mean, stdev

end

## TEST

array = [9,20,16,55,8,2,3,1,6,10,22,11,16,9,5,7,55,17]

mean,std = mean\_sigma(array)

print "Array: #{array}\n"

puts "Mean = #{mean}, \nStandard Deviation = #{std}"

OUTPUT:

Array: [9, 20, 16, 55, 8, 2, 3, 1, 6, 10, 22, 11, 16, 9, 5, 7, 55, 17]

Mean = 15.11111111111111,

Standard Deviation = 15.267572349004892

1. Answer:

#METHOD

def sort(values)

sorted\_values = Array.new(values)

(1...values.length).each do |i|

j = i

while j > 0 and sorted\_values[j-1] > sorted\_values[j]

sorted\_values[j], sorted\_values[j-1] = sorted\_values[j-1], sorted\_values[j]

j -= 1

end

end

return sorted\_values

end

#TEST

array = [9,20,16,55,8,2,3,1,6,10,22,11,16,9,5,7,55,17]

sorted\_array = sort(array)

puts "array"

array.each{|x| print "#{x} "}

puts

puts "sorted\_array"

sorted\_array.each{|x| print "#{x} "}

puts

puts "original array"

array.each{|x| print "#{x} "}

puts

OUTPUT:

array

9 20 16 55 8 2 3 1 6 10 22 11 16 9 5 7 55 17

sorted\_array

1 2 3 5 6 7 8 9 9 10 11 16 16 17 20 22 55 55

original array

9 20 16 55 8 2 3 1 6 10 22 11 16 9 5 7 55 17

1. Answer:

class Triangle

# CONSTRUCTOR

def initialize(x,y,z)

@x,@y,@z= x,y,z

@triangle = true;

end

# GETTERS AND SETTERS

attr\_accessor :x,:y,:z

# FIND PERIMETER

def perimeter

if @triangle == true

@x+@y+@z

else

print "Not Avilable."

end

end

def area

if @triangle == true

p = self.perimeter

phalf= p/2.0

return Math.sqrt(phalf\*(phalf-@x)\*(phalf-@y)\*(phalf-@z))

else

print "Not Avilable."

end

end

def test

if @x+@y < @z or @x+@z < @y or @y+@z < @x

@triangle = false

puts "a Non- Triangle Object."

else

if @x == @y && @y == @z

print "an Equilateral Triangle."

elsif (@x == @y && @x != @z) or (@y == @z && @x != @z) or (@x == @z && @y != @z)

print "an Isosceles Triangle."

elsif @x != @y && @x != @z && @z != @y

if (@x\*\*2).round == (@y\*\*2).round + (@z\*\*2).round or (@y\*\*2).round == (@x\*\*2).round + (@z\*\*2).round or (@z\*\*2).round == (@y\*\*2).round + (@x\*\*2).round

print "a Right Triangle."

else

print "a Scalene Triangle."

end

end

end

end

end

t1= Triangle.new(1,3,2)

print "Type: "

print"#{t1.test}\n"

print "Perimeter: "

print"#{t1.perimeter}\n"

print "Area: "

print"#{t1.area}\n\n"

t2= Triangle.new(3,3,2)

print "Type: "

print"#{t2.test}\n"

print "Perimeter: "

print"#{t2.perimeter}\n"

print "Area: "

print"#{t2.area}\n\n"

t3= Triangle.new(2,2,2)

print "Type: "

print"#{t3.test}\n"

print "Perimeter: "

print"#{t3.perimeter}\n"

print "Area: "

print"#{t3.area}\n\n"

t4= Triangle.new(1,2,1.73)

print "Type: "

print"#{t4.test}\n"

print "Perimeter: "

print"#{t4.perimeter}\n"

print "Area: "

print"#{t4.area}\n\n"

t5= Triangle.new(1,5,2)

print "Type: "

print"#{t5.test}"

print "Perimeter: "

print"#{t5.perimeter}\n"

print "Area: "

print"#{t5.area}\n\n"

OUTPUT:

Type: a Scalene Triangle.

Perimeter: 6

Area: 0.0

Type: an Isosceles Triangle.

Perimeter: 8

Area: 2.8284271247461903

Type: an Equilateral Triangle.

Perimeter: 6

Area: 1.7320508075688772

Type: a Right Triangle.

Perimeter: 4.73

Area: 0.8649981788275631

Type: a Non- Triangle Object.

Perimeter: Not Avilable.

Area: Not Avilable.

1. Answer:

class Array

def limited? (amin, amax)

self.each do |value|

unless amin <= value && value<= amax

return false

end

end

return true

end

def sorted?

# CHECK INCREASING SEQUENCE

self.each\_with\_index do |value, index|

if index!= self.length - 1

break unless value <= self[index+1]

else

return "+1"

end

end

# CHECK DECREASING SEQUENCE

self.each\_with\_index do |value, index|

if index!= self.length - 1

break unless value >= self[index+1]

else

return "-1"

end

end

return "0"

end

end

#TEST

numbers= [9,7,4,2,1]

numbers.limited?(2,12)

numbers.sorted?

puts "Array: #{numbers}"

puts "number.limited?(5,11): #{numbers.limited?(2,12)}"

puts "number.sorted?: #{numbers.sorted?}"

puts

numbers= [1, 2, 3, 4, 5, 6, 7, 8]

numbers.limited?(0,10)

numbers.sorted?

puts "Array: #{numbers}"

puts "number.limited?(5,11): #{numbers.limited?(0,10)}"

puts "number.sorted?: #{numbers.sorted?}"

puts

numbers= [9,2,4,8,10,11,12]

numbers.limited?(1,10)

numbers.sorted?

puts "Array: #{numbers}"

puts "number.limited?(5,11): #{numbers.limited?(1,10)}"

puts "number.sorted?: #{numbers.sorted?}"

puts

OUTPUT:

Array: [9, 7, 4, 2, 1]

number.limited?(5,11): false

number.sorted?: -1

Array: [1, 2, 3, 4, 5, 6, 7, 8]

number.limited?(5,11): true

number.sorted?: +1

Array: [9, 2, 4, 8, 10, 11, 12]

number.limited?(5,11): false

number.sorted?: 0