```
// This work is licensed under a Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA
4.0) https://creativecommons.org/licenses/by-nc-sa/4.0/
// © crazyrabbitheart

//@version=5
strategy("Trend strategy for Long", overlay = true, pyramiding = 1000, process_orders_on_close = true,
calc_on_every_tick = true, max_bars_back = 5000, max_lines_count = 500, max_labels_count = 500)

pivotLength = input.int(15, title = "pivot Length For Trend")
touchNum = input.int(3, title = "Touch Number")
valid = input.float(0.1, title = "valid percentage")
enablePivotToCheck = input.bool(false, title = "Enable Pivot To Valid")
plen = input.int(5, title = "pivot Length For valid")

isMulti = input.bool(true, title = "Enable Multi Trend")
posNum = input.int(1, title = "position number")

riskAmount = input.int(defval = 100, title = "Risk Amount", group = "strategy", minval = 1)
EnableContractSizeByDefault = input.bool(false, title = "Enable Default Contract Size")
setMethodForTP = input.string(defval = "RiskAwardRatio", title = "Set TP Method", options =
["RiskAwardRatio", "LookBackCandles", "Fibonacci"])
riskAwardRatio = input.float(defval = 1.5, title = "riskAwardRatio", group = "strategy", minval = 1.0)
lookBackCandles = input.int(defval = 10, title = "Look Back Candles", minval =1)
// nextCandles = input.int(1, title = "candles to check breakout")
sourceForTP = input.string(defval = "Close", title = "Source for TP", options = ["Close", "High/Low"])
enabledSL = input.bool(true, title = "Turn On/Off SL")
plenforsl = input.int(3, title = "pivot length for sl")
offsetForSL = input.float(0.0, title = "Buffer For SL", minval = 0.0)
enabledTrailing = input.bool(false, title = "Turn On/Off trailing stop")

lengthForFib = input.int(15, title = "Pivot length for Fibonacci", group = "Fibonacci")
// isFib1 = input.bool(true, title = "", group = "Fibonacci", inline = "Fibonacci1")
fibLevel1 = input.float(0.618, title = "Pivot length for Fibonacci", group = "Fibonacci", inline =
"Fibonacci1")
// isFib2 = input.bool(false, title = "", group = "Fibonacci", inline = "Fibonacci2")
fibLevel2 = input.float(1, title = "Pivot length for Fibonacci", group = "Fibonacci", inline = "Fibonacci2")
// isFib3 = input.bool(false, title = "", group = "Fibonacci", inline = "Fibonacci3")
fibLevel3 = input.float(1.312, title = "Pivot length for Fibonacci", group = "Fibonacci", inline =
"Fibonacci3")
// isFib4 = input.bool(false, title = "", group = "Fibonacci", inline = "Fibonacci4")
fibLevel4 = input.float(1.618, title = "Pivot length for Fibonacci", group = "Fibonacci", inline =
"Fibonacci4")

enabledSession = input.bool(false, title = "", group = "Session", inline = "Session")
i_sess = input.session("0900-1800", "Session", group = "Session", inline = "Session")

t = time(timeframe.period, i_sess)
bgcolor(time == t and enabledSession ? color.new(color.white, 95) : na)

ph = ta.pivothigh(high, pivotLength, pivotLength)
pl = ta.pivotlow(low, pivotLength, pivotLength)

var phh = 0.0
var pll = 0.0
```

```
phh := na(ph) ? phh[1] : ph     //pivot high value
pll := na(pl) ? pll[1] : pl     //pivot low value

var phbar = 0
var plbar = 0

phbar := na(ph) ? phbar[1] : bar_index - pivotLength    // pivot high bar_index
plbar := na(pl) ? plbar[1] : bar_index - pivotLength    // pivot low bar_index

phbar := na(phbar) ? 0 : phbar
plbar := na(plbar) ? 0 : plbar

pph = ta.pivothigh(high, plen, plen)    // pivot high for valid
ppl = ta.pivotlow(low, plen, plen)      // pivot low for valid

var pphbar = 0
var pplbar = 0

pphbar := na(pph) ? pphbar[1] : bar_index - plen    // pivot high bar_index for valid
pplbar := na(ppl) ? pplbar[1] : bar_index - plen    // pivot low bar_index for valid

h = ta.pivothigh(high, plenforsl, plenforsl)        //pivot high for SL
l = ta.pivotlow(low, plenforsl, plenforsl)          //pivot low for SL

var phsl = 0.
var plsl = 0.

phsl := na(h) ? phsl[1] : h
plsl := na(l) ? plsl[1] : l

h := ta.pivothigh(high, lengthForFib, lengthForFib)
l := ta.pivotlow(low, lengthForFib, lengthForFib)

var phFib = 0.
var plFib = 0.

phFib := na(h) ? phFib[1] : h
plFib := na(l) ? plFib[1] : l

if isMulti
    var line[] trendline = array.new_line()
    var int[] distance = array.new_int()
    var float[] stepY = array.new_float()
    var int[] startPointX = array.new_int()
    var float[] startPointY = array.new_float()

    if not na(ph) and phh < phh[1]
        dis = bar_index - phbar[1]
        step = (phh[1] - phh) / (phbar - phbar[1])
        y = phh[1] - step * (bar_index - phbar[1])

        n = 0
        for int i = 0 to dis
            pricen = y + i * step
            if high[i] > pricen
            n := n + 1
         else
            if (pricen - high[i]) < ((high[i] - low[i]) * valid / 100)
               n := n + 1
      isValidLine = n >= touchNum ? true : false

      if enablePivotToCheck
         isValidLine := isValidLine and (pphbar[bar_index - phbar] < phbar and pphbar[bar_index - phbar] >
phbar[1]) ? true : false

      if isValidLine
         array.push(trendline, line.new(phbar[1], phh[1], bar_index, y, xloc = xloc.bar_index))
         array.push(distance, dis)
         array.push(stepY, step)
         array.push(startPointX, phbar[1])
         array.push(startPointY, phh[1])

   var string[] entryId = array.new_string()
   var float[] slPerEntry = array.new_float()
   var string[] fibLevel = array.new_string()
   var float[] isFibTriggered = array.new_float()

   i = 0
   for tline in trendline
      y2 = array.get(startPointY, i) - array.get(stepY, i) * (bar_index - array.get(startPointX, i))
      line.set_xy2(tline, bar_index, y2)
      if close > y2 and ((timeframe.isintraday and time == t and enabledSession) or not
timeframe.isintraday or not enabledSession)
         if strategy.opentrades < posNum
            j = 0
            minl = plsl
            while close < plsl[j]
               j := j + 1
               minl := plsl[j]
               if j == 1000
                  minl := low
                  break

            minl := offsetForSL > 0 ? minl - offsetForSL : minl
            minl := low > minl ? minl : low
            qtyLong = math.floor((riskAmount / math.abs(close - minl)) / syminfo.pointvalue)
            if (not EnableContractSizeByDefault and qtyLong <= 1) or qtyLong < 0
               qtyLong := 0
            else if EnableContractSizeByDefault and qtyLong <= 1
               qtyLong := 1
            takeProfitLong = close + (close - minl) * riskAwardRatio
            entryId_ = "long"+str.tostring(bar_index)
            strategy.entry(entryId_, strategy.long, qtyLong)
            array.push(entryId, entryId_)
            // array.push(entryPointX, bar_index)

            sl = line.new(x1=bar_index, y1=minl, x2=bar_index + 10, y2=minl, color=color.red, width=2)

            if setMethodForTP == "Fibonacci"
               k = 1
               q = phFib

               while close > phFib[k]
                  k := k + 1
                  q := phFib[k]
                  if k == 1000
                      q := close
                      break
               // label.new(bar_index, high, text = str.tostring(q))
               l1 = (q - minl) * fibLevel1 + minl
               l2 = (q - minl) * fibLevel2 + minl
               l3 = (q - minl) * fibLevel3 + minl
               l4 = (q - minl) * fibLevel4 + minl

               fl1 = line.new(x1=bar_index, y1=l1, x2=bar_index + 10, y2=l1, color=color.green, width=1)
               fl2 = line.new(x1=bar_index, y1=l2, x2=bar_index + 10, y2=l2, color=color.green, width=1)
               fl3 = line.new(x1=bar_index, y1=l3, x2=bar_index + 10, y2=l3, color=color.green, width=1)
               fl4 = line.new(x1=bar_index, y1=l4, x2=bar_index + 10, y2=l4, color=color.green, width=1)

               strlvl = str.tostring(l1) + "," + str.tostring(l2) + "," + str.tostring(l3) + "," + str.tostring(l4)

               // if isFib1
               //     strlvl := strlvl + str.tostring(l1)
               // if isFib2
               //     strlvl := strlvl + "," + str.tostring(l2)
               // if isFib3
               //     strlvl := strlvl + "," + str.tostring(l3)
               // if isFib4
               //     strlvl := strlvl + "," + str.tostring(l4)
               array.push(fibLevel, strlvl)
               array.push(isFibTriggered, 0.0)

            if enabledSL
               if setMethodForTP == "RiskAwardRatio" or setMethodForTP == "Fibonacci"
                  strategy.exit("TP/SL" + str.tostring(bar_index), entryId_, stop=minl, limit=takeProfitLong)
               else if setMethodForTP == "LookBackCandles"
                  strategy.exit("TP/SL" + str.tostring(bar_index), entryId_, stop=minl)

            array.push(slPerEntry, minl)

         array.remove(trendline, i)
         array.remove(distance, i)
         array.remove(stepY, i)
         array.remove(startPointX, i)
         array.remove(startPointY, i)
      i := i + 1

   exitLongCondition = false
   exitShortCondition = false

   if sourceForTP == "Close"
      exitLongCondition := close == ta.lowest(close, lookBackCandles+1) and open > close
      exitShortCondition := close == ta.highest(close, lookBackCandles+1) and open < close
   else
      exitLongCondition := low == ta.lowest(low, lookBackCandles+1) and open > close
```

```
        exitShortCondition := high == ta.highest(high, lookBackCandles+1) and open < close

    if setMethodForTP == "LookBackCandles" //and bar_index > entryIndexh
        if (strategy.position_size > 0 and exitLongCondition) or (strategy.position_size < 0 and
exitShortCondition)
            strategy.close_all()
            array.clear(entryId)
            array.clear(slPerEntry)
            array.clear(fibLevel)
            array.clear(isFibTriggered)

    if setMethodForTP == "Fibonacci"
        i := 0
        for isTrig in isFibTriggered
            if isTrig > 0
                b = array.get(fibLevel, i)
                a = str.split(b, ",")
                if array.size(a) > 0
                    for k = 0 to array.size(a) - 1
                        if close > str.tonumber(array.get(a, k)) and close < isTrig and open < isTrig
                            strategy.close(array.get(entryId, i))
                            array.remove(entryId, i)
                            array.remove(slPerEntry, i)
                            array.remove(fibLevel , i)
                            array.remove(isFibTriggered, i)
                            break
            i := i + 1

        i := 0
        for fib in fibLevel
            a = str.split(fib, ",")
            if array.size(a) > 0
                for k = 0 to array.size(a) - 1
                    if close > str.tonumber(array.get(a, k))
                        // log.error(str.tostring(bar_index))
                        // log.info(array.get(a, k))
                        array.set(isFibTriggered, i, str.tonumber(array.get(a, k)))
            i := i + 1

    i := 0
    if enabledTrailing
        for slPerEntry_ in slPerEntry
            if plsl > slPerEntry_ and close < plsl
                strategy.close(array.get(entryId, i))
                array.remove(entryId, i)
                array.remove(slPerEntry, i)
                array.remove(fibLevel , i)
                array.remove(isFibTriggered, i)
            i := i + 1

    if na(t) and enabledSession
        strategy.close_all()
        array.clear(entryId)
        array.clear(slPerEntry)
        array.clear(fibLevel)
        array.clear(isFibTriggered)

else
    var line line1 = na
    var hvalid = false
    var distanceh = 0
    var steph = 0.
    var startLineXPointh = 0
    var startLineYPointh = 0.

    if not na(ph) and phh < phh[1] and not hvalid
        distanceh := bar_index - phbar[1]
        steph := (phh[1] - phh) / (phbar - phbar[1])
        y = phh[1] - steph * (bar_index - phbar[1])
        line1 := line.new(phbar[1], phh[1], bar_index, y, xloc = xloc.bar_index)
        startLineXPointh := phbar[1]
        startLineYPointh := phh[1]

        n = 0
        for int i = 0 to distanceh
            pricen = line.get_price(line1, bar_index - i)
            if high[i] > pricen
                n := n + 1
                // label.new(bar_index - i, high[i], text = "", style = label.style_circle, size = size.tiny)
            else
                if (pricen - high[i]) < ((high[i] - low[i]) * valid / 100)
                    n := n + 1
                    // label.new(bar_index - i, high[i], text = "", style = label.style_circle, size = size.tiny)
        hvalid := n >= touchNum ? true : false

        if enablePivotToCheck
            hvalid := hvalid and (pphbar[bar_index - phbar] < phbar and pphbar[bar_index - phbar] > phbar[1])
? true : false
    else
        distanceh := distanceh[1]
        steph := steph[1]
        startLineXPointh := startLineXPointh[1]
        startLineYPointh := startLineYPointh[1]
        hvalid := hvalid[1]

    var entryIndexh = 0
    var entrySl = 0.
    var fibLevel = ""
    var isFibTriggered = 0.

    if hvalid
        y2 = startLineYPointh - steph * (bar_index - startLineXPointh)
        line.set_xy2(line1, bar_index, y2)
        if close > y2 and ((timeframe.isintraday and time == t and enabledSession) or not
timeframe.isintraday or not enabledSession)
            if strategy.opentrades <= posNum
                j = 0
                minl = plsl
                while close < plsl[j]
                    j := j + 1
                    minl := plsl[j]
                    if j == 1000
                        minl := low
                        break
                minl := offsetForSL > 0 ? minl - offsetForSL : minl
                minl := low > minl ? minl : low
                qtyLong = math.floor((riskAmount / math.abs(close - minl)) / syminfo.pointvalue)
                if (not EnableContractSizeByDefault and qtyLong <= 1) or qtyLong < 0
                    qtyLong := 0
                else if EnableContractSizeByDefault and qtyLong <= 1
                    qtyLong := 1
                takeProfitLong = close + (close - minl) * riskAwardRatio
                strategy.entry("long"+str.tostring(bar_index), strategy.long, qtyLong)
                entrySl := minl
                log.error(str.tostring(bar_index))
                log.info(str.tostring(minl))
                // label.new(bar_index, high, text = str.tostring(qtyLong))
                line3 = line.new(x1=bar_index, y1=minl, x2=bar_index + 10, y2=minl, color=color.red, width=2)
                entryIndexh := bar_index

                if setMethodForTP == "Fibonacci"
                    k = 1
                    q = phFib

                    while close > phFib[k]
                        k := k + 1
                        q := phFib[k]
                        if k == 1000
                            q := close
                            break
                    // label.new(bar_index, high, text = str.tostring(q))
                    l1 = (q - minl) * fibLevel1 + minl
                    l2 = (q - minl) * fibLevel2 + minl
                    l3 = (q - minl) * fibLevel3 + minl
                    l4 = (q - minl) * fibLevel4 + minl

                    fl1 = line.new(x1=bar_index, y1=l1, x2=bar_index + 10, y2=l1, color=color.green, width=2)
                    fl2 = line.new(x1=bar_index, y1=l2, x2=bar_index + 10, y2=l2, color=color.green, width=2)
                    fl3 = line.new(x1=bar_index, y1=l3, x2=bar_index + 10, y2=l3, color=color.green, width=2)
                    fl4 = line.new(x1=bar_index, y1=l4, x2=bar_index + 10, y2=l4, color=color.green, width=2)

                    // fibLevel := isFib1?l1:isFib2?l2:isFib3?l3:isFib4?l4:0
                    fibLevel := str.tostring(l1) + "," + str.tostring(l2) + "," + str.tostring(l3) + "," + str.tostring(l4)
                    isFibTriggered := 0.

                if enabledSL
                    if setMethodForTP == "RiskAwardRatio" or setMethodForTP =="Fibonacci"
                        strategy.exit("TP/SL" + str.tostring(bar_index), "long"+str.tostring(bar_index), stop=minl,
limit=takeProfitLong)
                    else if setMethodForTP == "LookBackCandles"
                        strategy.exit("TP/SL" + str.tostring(bar_index), "long"+str.tostring(bar_index), stop=minl)
            hvalid := false

    exitLongCondition = false
    exitShortCondition = false
```

```
if sourceForTP == "Close"
    exitLongCondition := close == ta.lowest(close, lookBackCandles+1) and open > close
    exitShortCondition := close == ta.highest(close, lookBackCandles+1) and open < close
else
    exitLongCondition := low == ta.lowest(low, lookBackCandles+1) and open > close
    exitShortCondition := high == ta.highest(high, lookBackCandles+1) and open < close

if setMethodForTP == "LookBackCandles" and bar_index > entryIndexh
    if (strategy.position_size > 0 and exitLongCondition)
        strategy.close_all()

    if (strategy.position_size < 0 and exitShortCondition)
        strategy.close_all()

if setMethodForTP == "Fibonacci"
    if close < isFibTriggered and open < isFibTriggered
        strategy.close_all()


    a = str.split(fibLevel, ",")
    if array.size(a) > 0
        for k = 0 to array.size(a) - 1
            if close > str.tonumber(array.get(a, k))
                isFibTriggered := str.tonumber(array.get(a, k))

if enabledTrailing
    if plsl > entrySl and close < plsl
        strategy.close_all()

if na(t) and enabledSession
    strategy.close_all()
```