Gmail

JOOYONG CHOI <etlersadm@gmail.com>

(제목 없음)

JOOYONG CHOI <etlersadm@gmail.com> 2021년 4월 7일 오후 3:29
임시보관

```
import random, os, yaml, time
import pandas as pd
from datetime import datetime

now = datetime.now() # current date and time
run_hms = now.strftime("%H%M%S")
date_time = now.strftime("%m/%d/%Y, %H:%M:%S")

list_deal_history = []
# 설정 값 불러오기
with open('deal.yaml', encoding='utf-8') as stream:
    deal_info = yaml.safe_load(stream)
# 거래를 이어갈 파일 명칭
txt_name = deal_info["txt_name"]
# 최초 지갑
possesion = deal_info["possesion"]
# 매도 기준이 되는 수익률
profit_rate = deal_info["profit_rate"]
# 매수 기준이 되는 거래 틱 정보
tick_amount = deal_info["tick"]["tick_amount"]
tick_over = deal_info["tick"]["tick_over"]
tick_under = deal_info["tick"]["tick_under"]
# 추출 대기 초
wait_sec = deal_info["wait_sec"]
# 수행시간
start_hms = deal_info["run_hms"]["start_hms"]
stop_hms = deal_info["run_hms"]["stop_hms"]

# 전 거래내역 저장
def close_data(in_param):
    f = open(txt_name, "w")
    f.write(in_param)
    f.close()

# 텔레그램 메세지 전송
def send_message(div, deal_qty):
    pass

# 실시간 체결가
def get_agreement_price():
    agreement_price = 0
    # Cybos API를 통해서 추출
    return agreement_price

# 시장가로 매수 & 매도 전송
```

```
    def send_deal(div, deal_qty):
        send_message(div, deal_qty)
        pass

    # 실제 거래
    def execute(day_cnt):
        # 파일에 존재하는 경우 즉, 어제 매도가 안된 경우 매도를 위한 내역 추출
        try:
            f = open(txt_name, "r")
            data = f.read()
            last_price = int(data.split(" ")[0])
            buysell_qty = int(data.split(" ")[1])
            deal_amount = int(data.split(" ")[2])
            f.close()
        # 어제 마지막에 매도까지 한 경우
        except:
            last_price = 27675
            buysell_qty = 0
            deal_amount = possesion

        list_price = []
        std_price = last_price
        # 일 4천번 정도 거래내역이 있다고 가정. 추후 실시간 긁어오는 값으로 대체
        for _ in range(4000):
            std_price = std_price + random.randint(-2, 2) * 5
            list_price.append(std_price)
        # 초기값 설정
        day_profit = 0
        deal_cnt = 0
        bought_price = 0
        sell_price = 0
        div = 1
        idx = 0
        # 전일 매도 못한 경우로 매도로 설정
        if buysell_qty > 0:
            div = 2
            sell_price = last_price
            bought_price = last_price
        # 현재가 추출
        # 지정한 시간 동안 수행
        # while True:
        #     now = datetime.now() # current date and time
        #     run_hms = now.strftime("%H%M%S")
        #     # 시작 전이면 대기 메세지 출력하면서 대기
        #     if run_hms < start_hms:
        #         print("Waiting...", run_hms)
        #         continue
        #     # 시간이 지났으면 종료하러 나감
        #     if run_hms > stop_hms: break
        #     # 체결금액 추출
        #     now_price = get_agreement_price()
        #     # 대기
        #     time.sleep(wait_sec)
        for now_price in list_price:
            # 매도까지 끝난 경우의 종가
```

```
        end_price = now_price
        list_deal = []
        # 매수
        if div == 1:
            # 매도 마지막 금액으로 매수할 구간 설정
            price_over = last_price + (tick_amount * tick_over)
            price_under = last_price - (tick_amount * tick_under)
            # 매수 구간에 들어온 경우
            if (now_price > price_over or now_price < price_under):
                buysell_qty = int(deal_amount / now_price)
                send_deal(div, buysell_qty)
                list_deal.append(str(day_cnt + 1))
                list_deal.append("BUY")
                list_deal.append(now_price)
                list_deal.append(buysell_qty)
                list_deal.append(0)
                deal_cnt += 1
                div = 2
                bought_price = now_price
                sell_price = int(now_price + now_price * profit_rate)
        # 매도
        else:
            if now_price > sell_price:
                day_profit = day_profit + ((now_price * buysell_qty) - (bought_price * buysell_qty))
                send_deal(div, buysell_qty)
                list_deal.append(str(day_cnt + 1))
                list_deal.append("SELL")
                list_deal.append(now_price)
                list_deal.append(buysell_qty)
                list_deal.append(day_profit)
                deal_cnt += 1
                div = 1
                last_price = bought_price
                buysell_qty = 0
        # 거래내역이 존재하는 경우
        if len(list_deal) > 0:
            list_deal_history.append(list_deal)
        # 인덱스 증가
        idx += 1
    # 당일 매도 못한 경우 파일에 저장하여 다음 날에 매도 진행
    if bought_price > 0:
        close_data(str(bought_price) + " " + str(buysell_qty) + " " + str(deal_amount + day_profit))
    # 당일 매도까지 했다면 최종 가격을 종가로
    else:
        close_data(str(end_price) + " " + "0" + " " + str(deal_amount + day_profit))
    # 일 거래내역 회신
    return day_profit, deal_cnt, deal_amount
# 헤더 출력
print("#" * 50)
print("Day ", "Profit ", "deal ", " Rate", " Deal")
print("#" * 50)
# 전체 수익
profit_total = 0
# 일별 수행 및 결과 출력
for idx in range(120):
```

```
    profit, deal_cnt, deal_amount = execute(idx)
    profit_total += profit
    print(str(idx + 1).zfill(3) + " Day", profit_total, deal_amount, str(round(round(profit_total / possesion, 4) *
 100, 2)) + "%", deal_cnt)
# 사용했던 파일 삭제
os.remove(txt_name)
# 거래내역 CSV 파일로 저장
list_cols = [
    "Days", "DIV", "Bought", "Qty", "Profit"
]
df_deal = pd.DataFrame(list_deal_history, columns=list_cols)
df_deal.to_csv("deal_result.csv", index=False)


# 거래를 이어갈 파일 명칭
txt_name: "deal.txt"
# 지갑. 거래 시작 금액
possesion: 3000000
# 매도 기준이 되는 수익률
profit_rate: 0.012
# 매수 기준이 되는 거래 틱 정보
tick:
  tick_amount: 5
  tick_over: 3
  tick_under: 9
# 추출 대기초
wait_sec: 0.5
# 수행시간
run_hms:
  start_hms: "085959"
    stop_hms: "152959"
```

최주용 - CHOI JooYong
010.5000.6104
IT 프리랜서 이틀러스(ETLERS)

Cloud, BigData & RDB ETL(Talend, Informatica) 및 Python 개발
데이터 이행 및 검증 자동화 시스템 구축