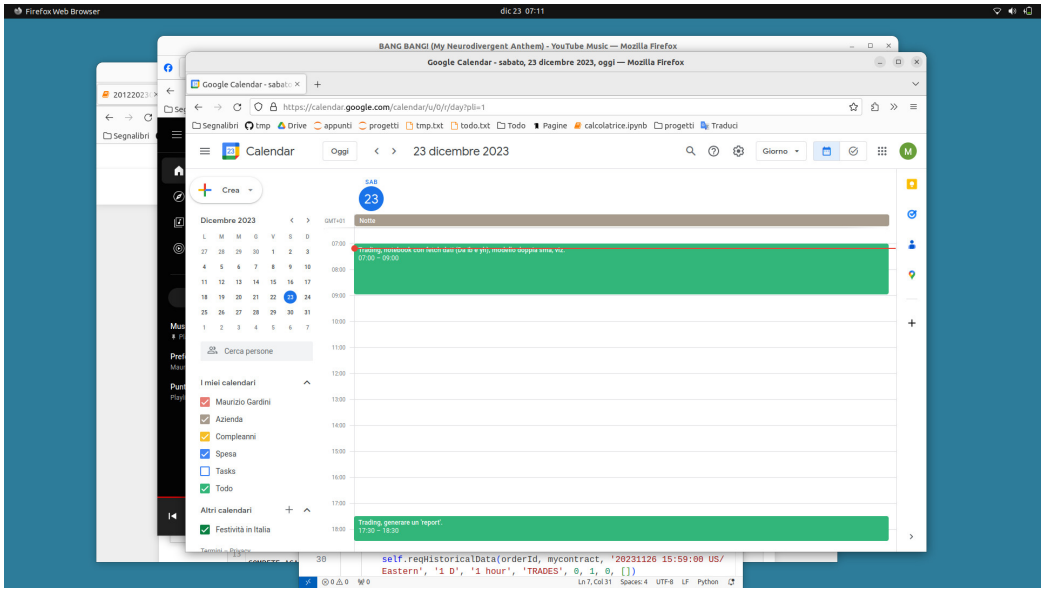

È unire questo:

```
# Codice per interrogare ib

from ibapi.client import *
from ibapi.wrapper import *

class IBapi(EWrapper, EClient):
    def __init__(self):
        EClient.__init__(self, self)
        self.data = []

    def nextValidId(self, orderId: int):

        mycontract = Contract()
        mycontract.symbol = "AAPL"
        mycontract.secType = "STK"
        mycontract.exchange = "SMART"
        mycontract.currency = "USD"
```

```
        self.reqHistoricalData(orderId, mycontract, '20231126
        ↪  15:59:00 US/Eastern', '1 D', '1 hour', 'TRADES', 0,
        ↪  1, 0, [])

    def historicalData(self, reqId, bar):
        self.data.append([bar.date, bar.close])
        app.disconnect()

app = IBapi()
app.connect('127.0.0.1', 7497, 100)
app.run()
```

Con questo:

```
# Codice per piazzare un ordine su ib

from ibapi.client import *
from ibapi.wrapper import *

class TestApp(EClient, EWrapper):
  def __init__(self):
    EClient.__init__(self, self)

  def nextValidId(self, orderId: OrderId):

    mycontract = Contract()
    mycontract.symbol = "AAPL"
    mycontract.secType = "STK"
    mycontract.exchange = "SMART"
    mycontract.currency = "USD"

    self.reqContractDetails(orderId, mycontract)

  def contractDetails(self, reqId: int, contractDetails:
  ↪  ContractDetails):
    print(contractDetails.contract)
```

```
        myorder = Order()
        myorder.orderId = reqId
        myorder.action = "BUY"
        myorder.tif = "GTC"
        myorder.orderType = "LMT"
        myorder.lmtPrice = 1440.80
        myorder.totalQuantity = 1

        self.placeOrder(myorder.orderId, contractDetails.contract,
        ↪  myorder)


    def openOrder(self, orderId: OrderId, contract: Contract,
    ↪  order: Order, orderState: OrderState):
        print(f"openOrder. orderId: {orderId}, contract:
        ↪  {contract}, order: {order}")

    def orderStatus(self, orderId: OrderId, status: str, filled:
    ↪  Decimal, remaining: Decimal, avgFillPrice: float, permId:
    ↪  int, parentId: int, lastFillPrice: float, clientId: int,
    ↪  whyHeld: str, mktCapPrice: float):
        print(f"orderId: {orderId}, status: {status}, filled:
        ↪  {filled}, remaining: {remaining}, avgFillPrice:
        ↪  {avgFillPrice}, permId: {permId}, parentId: {parentId},
        ↪  lastFillPrice: {lastFillPrice}, clientId: {clientId},
        ↪  whyHeld: {whyHeld}, mktCapPrice: {mktCapPrice}")

    def execDetails(self, reqId: int, contract: Contract,
    ↪  execution: Execution):
        print(f"reqId: {reqId}, contract: {contract}, execution:
        ↪  {execution}")

app = TestApp()
app.connect("127.0.0.1", 7497, 100)
app.run()
```

Voi come fareste? In sostanza si tratta di trovare le parti simili o da una o dall'altra parte, per toglierle, facendo qualche aggiustamento.

E poi mettere giù un modello tipo:

$$SMA = \frac{P_1 + P_2 + P_n}{n}$$

Per fare qualche previsione. Sta notte ho letto degli altri modi carini per implementarlo su dei libri (Doing math with python e il Python for finance).

E mettiamo tutto su un quaderno Google colab aggiungendo l'interrogazione di yh finance per chi non usa ib.

Ah! Come si avvicina Babbo Natale secondo il NORAD: