Bot de trading automatizado
Challenge técnico TSS

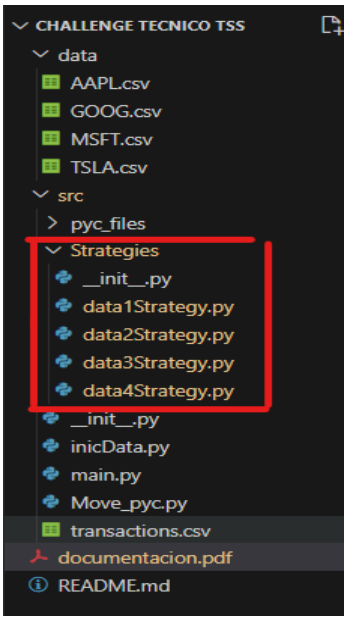
Arias Facundo

Para desarrollar el bot, se utilizó el framework “backtrader” con la versión de python 2.7.18.

Se utilizaron 4 sets de datos que corresponden a los activos GOOG, AAPL, MSFT y TSLA del año correspondiente a 2021.
Las estrategias utilizadas son:

- simple moving averages - SMA de 10 velas diarias
- simple moving averages - SMA de 30 velas diarias
- Golden cross pattern (con las sma descritas anteriormentes)

El código está planteado para agregar al cerebro del framework una estrategia por set de datos(activo), por eso se ve que se usa una estrategia por activo y luego se agregan las cuatro a dicho cerebro.




```
cerebro.addstrategy(data1Strategy)
cerebro.addstrategy(data2Strategy)
cerebro.addstrategy(data3Strategy)
cerebro.addstrategy(data4Strategy)
```

En cada estrategia se utilizaron los siguientes indicadores:

```
class data1Strategy(bt.Strategy):

    def __init__(self):

        # iniciando indicadores
        self.sma10 = bt.indicators.SimpleMovingAverage(self.datas[0], period=10, plotname="10 SMA_1")
        self.sma30 = bt.indicators.SimpleMovingAverage(self.datas[0], period=30, plotname="30 SMA_1")
        self.crossover = bt.indicators.CrossOver(self.sma10, self.sma30)

        #iniciando lista de ordenes de compra y venta
        self.buy_records = []
```

la lista “self.buy_records” se utiliza para registrar el historial de las órdenes de compra y ventas de los activos en la función “notify_order”

En cada estrategia se inicializan los siguientes datos:

```
# logica de data 1
portfolio_value = self.broker.getvalue()
cash_to_spend = portfolio_value * 0.10
price = self.datas[0].close[0]
size1 = cash_to_spend / price
position = self.getposition(self.datas[0])
```

- ***portafolio_value:*** dinero actual del portafolio
- ***cash_to_spend:*** dinero que se puede gastar en una posible compra (10% del valor del portafolio)
- ***price:*** precio de close del activo
- ***size:*** cantidad de activos que se puede comprar
- ***position:*** arreglo que contiene la cantidad del activo

La lógica para la compra y venta del activo utilizando el indicador sma con un periodo de 10 días es la siguiente (es igual con la sma con periodo de 30 dias):

```
# logica de sma10
strategy_id_sma10 = 'sma10'
if price > self.sma10[0]:
    if int(size1) > 0:
        self.buy(data=self.datas[0], size=int(size1), info={strategy_id_sma10})
elif price < self.sma10[0] and int(position.size) > 0:
    for elemento in self.buy_records:
        if elemento['atributo'] == {strategy_id_sma10}:
            self.sell(data=self.datas[0], size=int(elemento['clave'].pop()), info={strategy_id_sma10})
            self.buy_records.remove(elemento)
```

Si “*price*” (precio del activo) supera la sma de 10 realiza una orden de compra equivalente a la cantidad de activo que puede comprar, según el enunciado es el 10% del valor del portafolio. En el caso contrario, si la sma de 10 supera el precio del activo, se realiza la venta del mismo (en el caso de que se cuente con dicho activo) por una cantidad igual a la que se compro.

La lógica para el indicador crossover, es similar, la única diferencia es que se compara las sma de 10 y 30 velas diarias:

```
# logica de crossover
strategy_id_crossover = 'crossover'
if self.crossover > 0 :
    if int(size1) > 0:
        self.buy(data=self.datas[0], size=int(size1), info={strategy_id_crossover})
elif self.crossover < 0 and int(position.size) > 0:
    for elemento in self.buy_records:
        if elemento['atributo'] == {strategy_id_crossover}:
            self.sell(data=self.datas[0], size=int(elemento['clave'].pop()), info={strategy_id_crossover})
            self.buy_records.remove(elemento)
```

El archivo "*Move_pyc.py*" contiene un código para mover los archivos que terminan en .pyc que se generan cuando se usan clases. En resumen, ordena los archivos cache creados cuando se ejecuta el bot

El archivo "*inicData.py*" crea los set de datos y los agrega al cerebro

Por último, el historial de ordenes se guarda en un archivo llamado "*transactions.csv*"