Moduł wykresów – dokumentacja

Słowem wstępnym, oczywiście Python nie posiada czegoś takiego jak specyfikatory dostępu (to mnie przyznam dość zaskoczyło na początku), dlatego jeśli używam sformułowań „publiczne”, „prywatne” oznacza to „należy to traktować jako publiczne/prywatne”.

Jeśli nie napisano czy jakaś funkcja coś zwraca to nic nie zwraca.

Jeśli coś nie ma za sobą nawiasów to nie jest metodą, tylko polem.

Klasa ChartData

Klasa ta reprezentuje dane, które są rysowane na wykresie typu Chart lub CompareChart. Dane te tworzone są na podstawie obiektu klasy FinancialObject z modułu dataParser. Klasa ChartData odpowiada za pobranie odpowiedniego wycinka tych danych, transformację do formatu zrozumiałego dla funkcji matplotliba oraz za obliczanie wskaźników dla danych. W przypadku danych dla CompareChart przy okazji przekształcamy dane na procentowe zmiany od wartości odniesienia.

Konstruktor

__init__(finObj, start=None, end=None, step='monthly', compare=False)

finObj – obiekt klasy FinancialObject, z którego pobieramy dane

start, end – obiekty klasy datetime, które określają początkową i końcową datę

step – 'daily', 'weekly', 'monthly' w zależności od tego jakie chcemy zagęszczenie wartości

compare – czy dane mają być dla wykresu CompareChart (True) czy zwykłego Chart (False)

Metody i pola publiczne

open – lista z wartościami kursów otwarcia

close - lista z wartościami kursów zamknięcia

high - lista z wartościami maksymalnymi

low - lista z wartościami minimalnymi

volume – lista z wartościami wolumenu

date – lista obiektów klasy datetime (wartości osi x)

name – nazwa instrumentu finansowego (string)

quotes – lista krotek (time, open, close, high, low) – potrzebna, bo w tym formacie danych potrzebuje funkcja biblioteczna candlestick w wykresie

corrupted - True w przypadku wykrycia błędnych danych (tablice nierównej długości, start>=end), False jeśli wszystko ok

Metody i pola prywatne

fullArray – tablica przechowująca wszystkie (do poprawienia! Trzeba ustawić jakieś sensowne maksymalne duration do wskaźników np. 200 i pobierać tylko tyle ile naprawdę trzeba) wartości z FinancialObjecta. Potrzebne są one do obliczania wskaźników

getEarlierValues(length, type='close'): - Funkcja używana do wskaźników, które potrzebują wartości z okresu wcześniejszego niż dany okres (czyli chyba do wszystkich). Jeśli wcześniejsze wartości istnieją, są one pobierane z tablicy self.fullArray. W przeciwnym wypadku kopiujemy wartość początkową na wcześniejsze wartości (inaczej mowiąc, zakładamy, że przed debiotem kurs

spółki był stały).

length = ilość dodatkowych wartości, które potrzebujemy

zwraca listę o długości o length dłuższej niż pierwotnie miała odpowiednia tablica

metody o nazwach odpowiadających nazwom wskaźników (CCI, SMA, ...) - służą do pobrania wartości odpowiedniego wskaźnika z uwzględnieniem pobrania wcześniejszych danych. Zwracają one tablicę z obliczonymi wartościami wskaźnika o długości równej pozostałych tablic (date, open, low, ...)

Klasa Chart

Ta klasa odpowiada za podstawowy typ wykresu (z dynamicznie zmieniającymi się wskaźnikami, oscylatorem, wolumenem, rysowaniem po wykresie itd.). Dziedziczy po klasie FigureCanvas (biblioteczna z matplotliba), która to z kolei może być użyta jako widget Qt.

Konstruktor

__init__(parent, finObj=None, width=8, height=6, dpi=100) -Tworzy domyślny wykres (liniowy z wolumenem, bez wskaźników) dla podanych danych. Domyślny rozmiar to 800x600 pixli

parent – to, w czym umieszczamy wykres (pytajcie Dawida, on się zna na Qt)

finObj – obiekt klasy FinancialObject, który ma być reprezentowany na wykresie

width, height – wysokość/szerokość w calach

dpi – ilość pixli na cal. Zatem wymiar wykresu to width*dpi na height*dpi. Chociaż to i tak nie ma znaczenia, bo wykres dopasowuje się do parenta (chyba, cokolwiek znaczy QtGui.QSizePolicy.Expanding)

Metody i pola publiczne

setData(finObj, start=None, end=None, step='daily') – ustawia model danych dla wykresu, po czym go odświeża. Parametry – patrz dokumentacja ChartData

getData() – pobiera obiekt klasy ChartData dla aktualnego wykresu

setGrid(grid) - włącza lub wyłącza rysowanie grida

grid – True (grid on) / False (grid off)

setMainType(type) - Ustawiamy typ głównego wykresu

type – typ wykresu jako string ('point','line','candlestick','none')

updatePlot() – odświeża wszystkie podwykresy

addVolumeBars() – dodaje wolumen do wykresu

rmVolumeBars() – ukrywa wolumen

setScaleType(type) - Ustawia skalę liniową lub logarytmiczną na głównym wykresie.

type – 'linear' lub 'log'.

setMainIndicator(type): - ustawiamy, jaki wskaźnik chcemy wyświetlać na głównym wykresie

type - 'SMA', 'WMA, 'EMA', 'bollinger', dowolna inna wartość oznacza żaden

setOscPlot(type) – ustawiamy, jaki oscylator chcemy wyświetlać

type – 'momentum', 'CCI', 'RSI', 'ROC', 'williams', dowolna inna wartość oznacza żaden

setDrawingMode(mode) – określamy, czy można rysować na wykresie

mode – True = można rysować, False = nie można rysować

drawLine(x0,y0,x1,y1) – dodaje do wykresu dodatkową linię (do zaznaczania trendów i formacji)

x0, y0 współrzędne (na wykresie) początku linii

x1, y1 współrzędne (na wykresie) końca linii

clearLines(self) – czyści wszystkie dodatkowe linie z wykresu (trendy, formacje i to co sobie user ręcznie namalował)

drawRectangle(x, y, width, height, colour='blue', lwidth = 2.0, lstyle = 'dashed') – rysuje prostokąt na wykresie (używane do zaznaczania formacji świecowych i luk)

x, y, width, height – wymiary jako współrzędne **na wykresie** a nie na ekranie

reszta - http://matplotlib.sourceforge.net/api/artist_api.html#matplotlib.patches.Rectangle

clearRectangles(self) – usuwa prostokąty narysowane na wykresie

Metody i pola prywatne

data - obiekt klasy ChartData przechowujący dane

fig - rysowany wykres (tzn. obiekt klasy matplotlib.Figure)

mainPlot – obiekt klasy matplotlib.Axes reprezentujący główny wykres (kurs akcji w czasie + ewentualne średnie kroczące)

volumeBars - obiekt klasy matplotlib.Axes reprezentujący wykres wolumenu

oscPlot - obiekt klasy matplotlib.Axes reprezentujący wykres oscylatora

additionalLines - lista linii narysowanych dodatkowo na wykresie (przez usera, albo przez wykrycie trendu/formacji)

rectangles - lista prostokątów narysowanych dodatkowo na wykresie (do zaznaczania formacji świecowych oraz luk)

mainType - typ głównego wykresu, string mogący przyjąć wartości: 'line', 'point', 'candlestick'. W przypadku podania innej wartości wykres się nie narysuje

oscType - typ oscylatora, string mogący przyjąć wartości: 'momentum', 'CCI', 'ROC', 'RSI', 'williams'. W przypadku podania innej wartości oscylator się nie narysuje

mainIndicator - typ wskaźnika rysowany dodatkowo na głównym wykresie (tzn. średnia krocząca). Przyjmuje następujące wartości: 'SMA', 'WMA', 'EMA', 'bollinger'. W przypadku podania innej wartości wskaźnik się nie narysuje.

x0, y0 - współrzędne pierwszego kliknięcia w trybie rysowania linii na wykresie

drawingMode – flaga boolowska określająca, czy tryb rysowania po wykresie jest włączony czy nie.

scaleType - rodzaj skali na osi y, w przypadku podania 'log' skala będzie logarytmiczna, natomiast dowolna inna wartość (preferowana 'linear') sprawi, że skala będzie liniowa

grid – flaga boolowska określająca, czy rysujemy grida

num_ticks – stała określająca ile etykiet jest rysowanych pod wykresem

Poniższe stałe przyjmują wartości od 0 do 1, jako ułamek wymiaru obiektu klasy Figure (tzn. wartość 0.2 oznacza 20% szerokości/wysokości)

margin – margines oddzielający wykres od krawędzi widgetu

maxSize – maksymalna wysokość/szerokość wykresu 2*margin+maxSize = 1

volHeight – wysokość wykresu wolumenu

oscHeight – wysokość wykresu oscylatora

addMainPlot() - metoda tworząca główny wykres (mainPlot). Wywoływana w konstruktorze.

updateMainPlot() - metoda odrysowująca główny wykres. Oprócz samego wywołania matplotlibowego plot() odpowiada także za skalowanie osi, wyświetlenie legendy. Wywoływana przez updatePlot(). W przypadku błędnych danych (self.data.corrupted==True) wykres nie zostanie odrysowany.

updateVolumeBars() - metoda odrysowująca wykres wolumenu. Działanie analogiczne do updateMainPlot. Wywoływane w updatePlot.

drawCandlePlot() - metoda odrysowująca główny wykres w postaci świecowej, korzysta z bibliotecznej funkcji matplotlib.finance.candlestick. Wywoływana przez updateMainPlot

updateMainIndicator() - metoda odrysowująca średnią kroczącą na głównym wykresie. Wywoływane przez updateMainPlot

updateOscPlot() - odrysowanie wykresu oscylatora – działanie analogiczne do updateMainPlot i updateVolumeBars. Wywoływane w updatePlot

fixOscLabels() - dobiera odpowiedni zakres wartości dla danego oscylatora i przenosi etykiety na osi y oscylatora na prawą stronę tak, żeby nie nachodziły na etykiety y mainPlota. Wywoływane przez updateOscPlot

formatDateAxis(ax) – formatuje etykiety osi czasu dla podanego jako parametr obiektu klasy matplotlib.Axes. Formatowanie polega na rozmieszczeniu etykiet, przypisaniu ich do dat, oraz wypisania w formacie YYYY-MM-DD. Ponadto ustawiamy rozmiar i wyrównanie do środka.

fixTimeLabels() - ukrywa etykiety czasu pod wszystkimi wykresami, z wyjątkiem najniższego (tzn. jeśli mamy narysowany duży wykres i oscylator, to etykiety z datami będą tylko pod oscylatorem). Wywoływane przez każdą z metod, która dodaje/usuwa któryś z podwykresów

fixPositions() - Dopasowuje pozycje i rozmiary podwykresów (głównego, wolumenu i oscylatora) w taki sposób, żeby zawsze wykorzystana była optymalna przestrzeń (tzn. żeby wykres miał zawsze wysokość maxSize). Wysokości oscPlota i volumeBarsów są stałe, natomiast mainPlot kurczy się i rozszerza tak, by zrobić im miejsce. Wywoływane zawsze w tym samym kontekście co fixTimeLabels

onClick() - listener do eventów myszy w trybie rysowania. Jego działanie polega na rysowaniu linii pomiędzy dwoma kolejnymi kliknięciami.

Klasa CompareChart

Jest to klasa reprezentująca wykres porównujący kilka instrumentów finansowych na jednym wykresie. Jest to dość okrojona wersja zwykłego Chart-a (nie mamy wolumenu, wskaźników, oscylatorów, jedyny typ do wyboru to liniowy). Z tego też względu powtarzające się atrybuty pozostaną bez komentarza.

Konstruktor

__init__(parent, width=8, height=6, dpi=100) – patrz konstruktor Charta, jedyna różnica jest taka, że nie podajemy na starcie danych (żeby cokolwiek narysować trzeba najpierw wywołać setData)

Metody i pola publiczne

setData(data, start=None, end=None, step='monthly') – metoda ustawiająca model danych rysowanych na wykresie. Jednocześnie weryfikowana jest ich poprawność (jeśli z którymkolwiek

elementem listy data jest coś nie tak, bądź w wyniku otrzymamy dane różnej długości, za self.data zostanie przypisana pusta lista)

data – lista obiektów klasy FinancialObject

pozostałe parametry – jak w setData do Charta

Poniższe działają dokładnie tak jak ich odpowiedniki w Chart

updatePlot(), setDrawingMode(), setScaleType(), drawLine(), clearLines()

Metody i pola prywatne

Data - lista obiektów klasy ChartData. Domyślnie (lub w przypadku błędnych danych) pusta

lineColors – lista kolorów linii dla kolejnych instrumentów. W przypadku rysowania większej ilości instrumentów niż długość tej tablicy (5) brane jest modulo i kolory się powtarzają.

fig, mainPlot, additionalLines, x0, y0, drawingMode, scaleType, num_ticks, margin, maxSize, formatDateAxis(ax), onClick() – patrz Chart

Klasa LightweightChart

Jest to klasa przeznaczona do rysowania „lekkich” wykresów. Poprzez lekkie mam na myśli po pierwsze to, że nie są powiązane z żadnymi wskaźnikami, oscylatorami, wolumenem, nie można po nich rysować etc. Po drugie, nie korzystają one z żadnej klasy pośredniczącej – podajemy po prostu dwie tablice: z wartościami osi czasu oraz z wartościami osi y. Zastosowanie – do rysowania wskaźników szerokości rynku (A/D Line, McClellan Oscillator, TRIN), indeksów, walut etc.

Konstruktor

__init__(parent, dates=None, values=None, name=None, width=3.2, height=2.4, dpi=100)

dates – lista stringów (nie żadne datetimesy, zwykłe stringi) z datami

values – lista wartości na osi y. Oczywiście musi byc tej samej długości co dates

name – nazwa, która będzie wyświetlona na etykiecie

pozostałe parametry – jak w Charcie

Metody i pola publiczne

setData(dates, values, name=None) – ustawiamy model danych do wykresu

parametry – patrz konstruktor

updatePlot() - odświeża i odrysowuje wykres

Metody i pola prywatne

fig, margin, maxSize, num_ticks – patrz Chart

plot – odpowiednik mainPlot w Chart

formatDateAxis() - spelnia tą samą funkcję co w Chart i CompareChart