Εθνικο Μετσοβιο Πολυτεχνειο

Προγραμματιστικά Εργαλεία και Τεχνολογίες για Επιστήμη Δεδομένων,

Χειμερινό Εξάμηνο 2021-2022

Εργασία Εξαμήνου

Σταύρος Κετσέας - ΑΜ: 034 00 084
email: stavrosketsсas@mail.ntua.gr, ΔΠΜΣ $\varepsilon.\delta\varepsilon.\mu^2$

Δομή Εργασίας

Η εργασία υλοποιείται από τρία βασικά python αρχεία:

- stockSonder.py
- Stock.py
- Transactions.py

Τα δύο τελευταία αποτελούν τις βιβλιοθήκες που χρειάζεται η κεντρική συνάρτηση stockSonder.py για την εύρεση και τον υπολογισμό της ζητούμενης ακολουθίας. Στο αρχείο Stock.py βρίσκεται η βασική κλάση στην οποία αποθηκεύουμε όλα τα στοιχεία τα οποία λαμβάνουμε από τα διαθέσιμα αρχεία για τις τιμές των μετοχών. Ενώ το αρχείο Transactions.py περιλαμβάνει τις απαραίτητες συναρτήσεις για την αναζήτηση τιμών, μετοχών και την υλοποίηση της βασικής λογικής για την εύρεση της λύσης.

Για την εκτέλεση του προγράμματος αφήνουμε τα αρχεία στην ιεραρχία ως έχουν. Στην συνέχεια στο αρχικό directory (όπου βρίσκεται και το αρχείο stockSonder.py) προσθέτουμε έναν φάκελο με όνομα «Stocks», ο οποίος πρέπει να περιλαμβάνει όλα τα txt αρχεία με τα στοιχεία των μετοχών.

Λογική Σκέψη

Η λογική κατά την οποία αντιμετωπίστηκε η εργασία αφορά κυρίως τα IntraDayTransactions. Μην γνωρίζοντας αν η χαμηλότερη τιμή (low) μίας μετοχής είναι πριν ή μετά της υψηλότερης (high) οδηγούμαστε στην καθημερινή εναλλαγή, αγορά και πώληση μετοχών.

Για κάθε μέρα κατασκευάζουμε έναν πίνακα με τα δεδομένα όλων των μετοχών. Για κάθε μέρα υπολογίζουμε τα πιθανά κέρδη που μπορεί να έχουμε αν αγοράσουμε μία μετοχή κατά το άνοιγμα (open) και την πώλησή της στην υψηλότερη τιμή ή στο κλείσιμο. Αποθηκεύουμε όλα αυτά τα δεδομένα σε έναν πίνακα και τον οποίο κάνουμε ταξινόμηση βάση του κέρδους (profit) που μπορεί κάθε μία να μας επιφέρει. Στην συνέχεια επιλέγουμε και αγοράζουμε την καλύτερη από αυτές. Για την αγορά ελέγχουμε ότι έχουμε τα χρήματα και ότι επίσης η αγορά της δεν θα ξεπεράσει το 10% του όγκου (volume) της ημερήσιας αγοράς εκείνη την ημέρα. Επομένως σε κάθε διαφορετική μέρα έχουμε την αγοροπωλησία μίας μετοχής, βάση των παραπάνω στοιχείων.

Αν για κάποια από τις ημέρες καμία μετοχή δεν επιφέρει κέρδος, τότε δεν εκτελούμε καμία εντολή, και επομένως μπορεί να υπάρχουν μέρες κατά τις οποίες δεν θα υπάρξει καμία αγοροπωλησία μετοχών. Αν και η παραπάνω λογική μπορεί να οδηγήσει στο να χάσουμε πιθανά κέρδη από μία μακροπρόθεσμη επένδυση, εγγυάται ότι συνεχώς θα έχουμε κέρδη μόνο.

Στην συνέχεια προστέθηκε η λογική σύγκρισης των ημερήσιων κερδών με την αγορά και κράτηση μίας μετοχής για ένα μεγαλύτερο διάστημα. Σε περίπτωση που η μακροχρόνια αγορά μίας μετοχής για μία περίοδο οδηγεί σε μεγαλύτερο κέρδος θα επιλέξουμε αυτή, έναντι του intra day transaction.

StockSonder.py

Το αρχείο αφορά την βασική main συνάρτηση του προβλήματος. Εδώ εισάγουμε τις απαραίτητες βιβλιοθήκες, διαβάζουμε τα στοιχεία από τα δοθέντα αρχεία και τα εκχωρούμε σε έναν πίνακα

για να διαχειριστούμε στην συνέχεια. Ορίζουμε το αρχικό μας κεφάλαιο (1$ κατά την άσκηση – μεταβλητή **money**) καθώς και τις ημέρες τις οποίες θέλουμε να μελετήσουμε για κάθε μετοχή, σε περίπτωση που επιθυμούμε την επιλογή μικρότερου σετ πληροφοριών (μεταβλητή **maxDays**). Σε αυτή, αν τεθεί ίσο με -1 τότε ελέγχουμε ολόκληρο το set των δεδομένων.

Μόλις κατασκευαστεί ο πίνακας τον δεδομένων των μετοχών προχωράμε στον υπολογισμό των Transactions, με την λογική που αναλύθηκε στο προηγούμενο κεφάλαιο. Στο τέλος εμφανίζουμε το κέρδος που θα έχουμε με βάση τα επιλεγμένα κριτήρια.

Τέλος, όλα τα αποτελέσματα και το σύνολο των συναλλαγών αποθηκεύονται σε ένα txt αρχείο στην ζητούμενη μορφή τους. Το αρχείο είναι έτοιμο για εισαγωγή στην ιστοσελίδα ελέγχου.

Stock.py

Αφορά το αρχείο που αποθηκεύει την κλάση για αναπαράσταση κάθε μετοχής στον κώδικα. Η κλάση αποτελείται από τα εξής βασικά χαρακτηριστικά:

- **Stock:** Όνομα μετοχής
- **Data:** Ο πίνακας με τα dictionaries με τα στοιχεία κάθε ημέρας όπως λαμβάνονται από το αντίστοιχο αρχείο
- **bestTransation:** Ο πίνακας που αποθηκεύει τα δεδομένα της πιο κερδοφόρας ημέρας για όλα τα δεδομένα της παρούσας μετοχής

Οι μέθοδοι της κλάσης αφορούν:

- **addData:** Κατά την ανάγνωση του αρχείου προσθέτει τα αναγνωσθέντα δεδομένα από το αρχείο, γραμμή κατά γραμμή, και τα αποθηκεύει στον πίνακα data. Τα αποθηκεύει ως dictionary
- **getIntraDayWin:** Υπολογίζει για μία συγκεκριμένη ημερομηνία το πιθανό κέρδος που μπορεί να επιφέρει η μετοχή
- **getProfitPeriod:** Υπολογίζει για μία συγκεκριμένη περίοδο το πιθανό κέρδος που μπορεί να επιφέρει η συγκεκριμένη μετοχή.
- **getBiggestDayAccimulation:** Επιστρέφει τα δεδομένα της ημέρας με το μεγαλύτερο κέρδος. Αποθηκεύει το αποτέλεσμα στην μεταβλητή bestTransaction.
- **getOpenPrice:** Επιστρέφει για μία ημέρα την τιμή open της μεταβλητής.
- **getVolume:** Για μία ημέρα επιστρέφει τον όγκο των συναλλαγών
- **buyAmmount:** Βάση ενός ποσού επιστρέφει για μία ημέρα τον αριθμό των μετοχών που μπορούν να αγοραστούν. Το σύνολο των μετοχών δεν πρέπει να ξεπερνάει σε κόστος τα χρήματα που διατίθενται καθώς επίσης και το 10% του συνολικού όγκου.

Τέλος στο ίδιο αρχείο έχουμε και την γενική συνάρτηση **readData** η οποία για τον δοθέν φάκελο θα διαβάσει όλα τα αρχεία σε αυτόν , για όλες τις μετοχές, και θα επιστρέψει τον πίνακα με τις κλάσεις των αντίστοιχων μετοχών. Η μεταβλητή **maxDays** αν τεθεί σε -1 τότε αναζητούνται όλα τα δεδομένα, διαφορετικά αναλύουμε μόνο τις maxDays ημέρες για κάθε διαφορετική μετοχή. Η συνάρτηση επιστρέφει τα δεδομένα σε ημερολογιακή διάταξη.

Transactions.py

Η βιβλιοθήκη υλοποιεί τις απαραίτητες συναρτήσεις για τον υπολογισμό των απαραίτητων στοιχείων και την τελική επιλογή των απαιτούμενων βημάτων για επίτευξη του στόχου. Οι συναρτήσεις αναφέρονται στις:

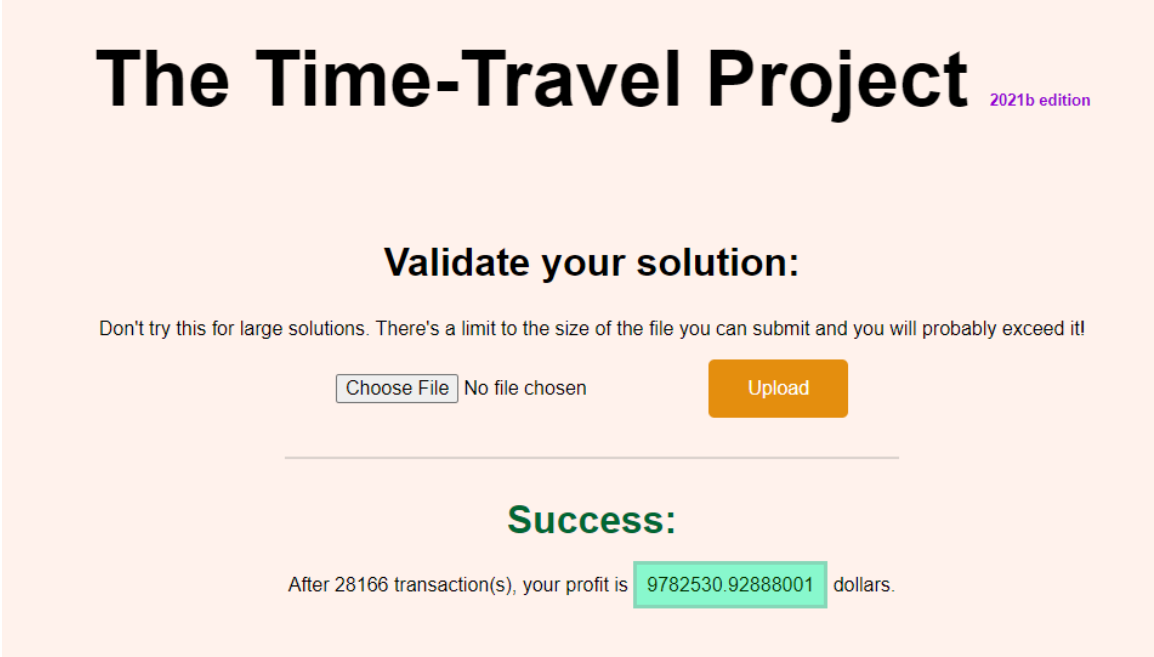
- **findBestSellOnDay:** Δεχόμενη τον πίνακα των μετοχών και μία ημερομηνία υπολογίζει και επιστρέφει την μετοχή που προσφέρει το μεγαλύτερο κέρδος, με τα σχετικά στοιχεία.
- **findBestSellOnPeriod:** Δεχόμενη τον πίνακα των μετοχών και μία περίοδο υπολογίζει και επιστρέφει την μετοχή που προσφέρει το μεγαλύτερο κέρδος σε αυτή την περίοδο δεχόμενη ως ημέρα αγοράς την αρχή της περιόδου και ως ημέρα πώλησης το τέλος της περιόδου.
- **findBestSellOnPeriodSmart:** Δεχόμενη τον πίνακα των μετοχών και μία περίοδο υπολογίζει και επιστρέφει την μετοχή που προσφέρει το μεγαλύτερο κέρδος σε αυτή την περίοδο, με τα σχετικά στοιχεία. Η ημέρα αγοράς είναι η τρέχουσα ενώ υπολογίζει την μελλοντική ημέρα κατά την οποία θα έχουμε το μεγαλύτερο κέρδος.
- **findBestIncreaseDaily:** Δεχόμενη τον πίνακα των μετοχών επιστρέφει την μετοχή με το μεγαλύτερο κέρδος σε όλο το διάστημα (ημερήσιο).
- **getAllBestIncreaseDaily:** Υπολογίζει για όλες τις μετοχές το μεγαλύτερο ημερήσιο κέρδος. Τις ταξινομεί βάση αυτού και επιστρέφει το αποτέλεσμα
- **becomeRich:** Δεχόμενη τον πίνακα των μετοχών, το σύνολο των ημερών και τα διαθέσιμα χρήματα, Υπολογίζει το σύνολο των συναλλαγών για το μέγιστο κέρδος. Για κάθε ημέρα υπολογίζουμε την καλύτερη μετοχή, με το μεγαλύτερο κέρδος, από αυτές που μπορούμε να αγοράσουμε βάση χρημάτων και όγκου συναλλαγών, και εκτελούμε την αγοροπωλησία της. Αν δεν υπάρχει κερδοφόρα μετοχή αγνοούμε την ημέρα. Εδώ, καθώς κάθε ημέρα αγοράζουμε και πουλάμε, πάντα αγοράζουμε 1 μετοχή. Επιστρέφει τον πίνακα των συναλλαγών καθώς και το κέρδος.
- **hodl:** Για μία περίοδο και τα διαθέσιμα χρήματα υπολογίζει και επιστρέφει την καλύτερη διαθέσιμη μετοχή για μακροχρόνια (βάση περιόδου) αγορά, ως προς το κέρδος.
- **buyStock:** Δεχόμενη μία μετοχή, τα δεδομένα αγοράς της μετοχής (ημερομηνία, open,..) τα διαθέσιμα χρήματα, και τον αριθμό συναλλαγών που μας απομένουν την ημέρα εκείνη, επιστρέφει την εντολή συναλλαγής. Αφορά την αγορά επομένως μίας μετοχής.
- **becomeRicher:** Συνάρτηση παρόμοια με την becomeRich. Σε αυτή την συνάρτηση ωστόσο αποφασίζουμε να κάνουμε και μακροχρόνιες αγορές. Θέτουμε ένα threshold κατά το οποίο ορίζει τα επιθυμητά κέρδη. Κάθε φορά που ξεπερνάμε αυτό το όριο χρημάτων στον λογαριασμό μας (1000 *3 για κάθε step), δεσμεύομε ένα υποσύνολο του ποσού μας για μακροχρόνιες αγορές (20%). Για αυτές ορίζουμε ένα σύνολο ημερών, περίοδος για κράτημα μετοχών (**hold_period).** Αν η μεταβλητή τεθεί σε -1 ελέγχουμε ως το ημερολογιακό τέλος. Για κάθε μέρα στην συνέχεια συγκρίνουμε τα κέρδη που θα μας επιφέρει ένα ημερήσιο συνάλλαγμα έναντι του κρατήματος αυτής της μετοχής. Αν το κέρδος για το κράτημα είναι **hold_threshold**(=3) φορές μεγαλύτερο από το ημερήσιο κέρδος, αποφασίζουμε να αγοράσουμε αυτή την μετοχή. Στην συνέχεια ελέγχουμε αν έχουμε άλλες διαθέσιμες συναλλαγές και εκτελούμε παρόμοια διαδικασία. Στο τέλος κάθε ημέρας ελέγχουμε μήπως κάποιες μακροχρόνιες αγορές μετοχών έχουν φτάσει στο τέλος της περιόδους τους (όπως έχει ήδη υπολογιστεί) και πρέπει να

πουληθούν. Για όλες τις παραπάνω διαδικασίες η μεταβλητή **portfolio** αποθηκεύει ως dictionary την κατάσταση των μετοχών που έχουμε. Στο τέλος της περιόδου πουλάμε όσες μετοχές έχουν μείνει σε αυτό. Η μεταβλητή **transactions** αποθηκεύει το σύνολο των συναλλαγών.

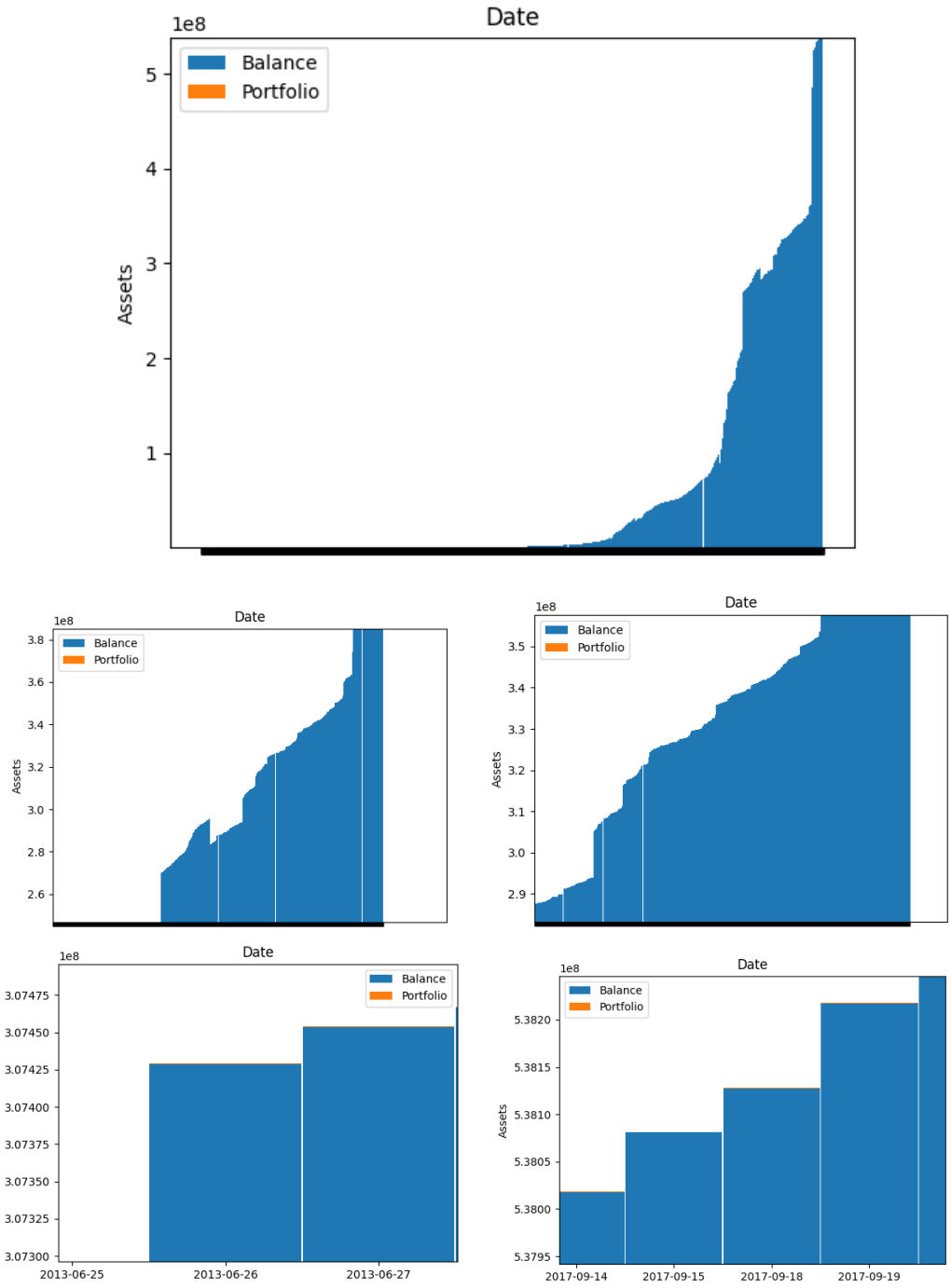
Αποτελέσματα

Το πρόγραμμα εκτελεί όλες τις απαραίτητες λειτουργίες και εμφανίζει αποτελέσματα. Τα αποτελέσματα, ακόμα και για την ημερήσια αγοροπωλησία μετοχών είναι αρκετά σημαντικά και για το σύνολο των δεδομένων προσφέρουν κέρδος έως και 9 με 10 εκατομμύρια. Παραδείγματα εμφανίζονται παρακάτω.

- Ημερήσιες αγοροπωλησίες για όλα τα δεδομένα:



Στο παράδειγμα αυτό εμφανίζουμε και το αντίστοιχο ζητούμενο γράφημα, κάνοντας μεγέθυνση σε τυχαίες ημερομηνίες. Πρόκειται για “stacked bar chart”, που απεικονίζει σε κάθε χρονική στιγμή το balance που έχουμε και τον αριθμό σε assets του portfolio μας.



Στις παραπάνω εικόνες παρατηρούμε ότι η κράτηση μίας μετοχής είναι αρκετά μικρή, καθώς η λογική μας βασίζεται σε ημερήσιες συναλλαγές. Συνεπώς δεν έχουμε μεγάλο όγκο στο portfolio. Για την κατασκευή του γραφήματος χρησιμοποιήθηκε η βιβλιοθήκη pyplot.

- Ημερήσιες συναλλαγές για 1000 μέγιστες συναλλαγές (small.txt) – αφορά τις πρώτες 1000 που εκτελούνται. Συνεπώς το κέρδος είναι μικρό.

The Time-Travel Project 2021b edition

Validate your solution:

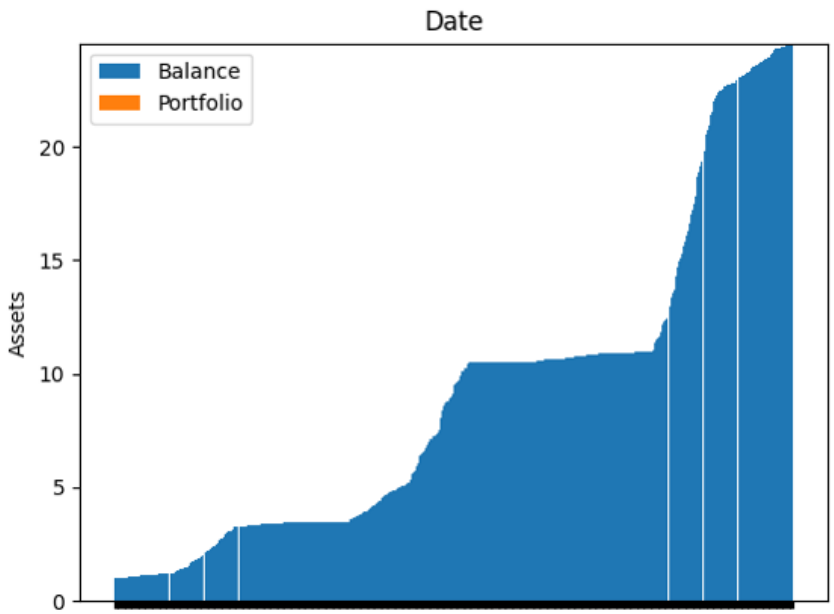
Don't try this for large solutions. There's a limit to the size of the file you can submit and you will probably exceed it!

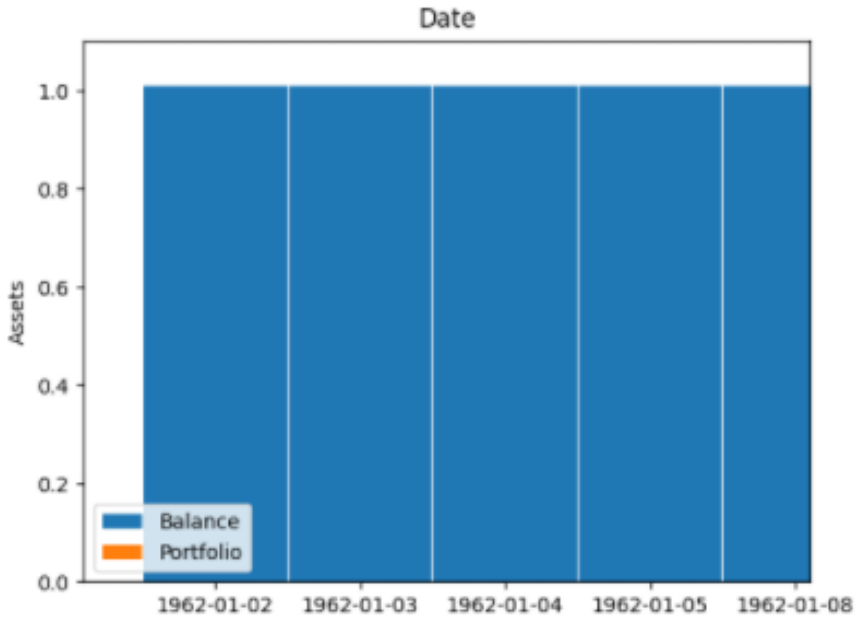
Choose File No file chosen Upload

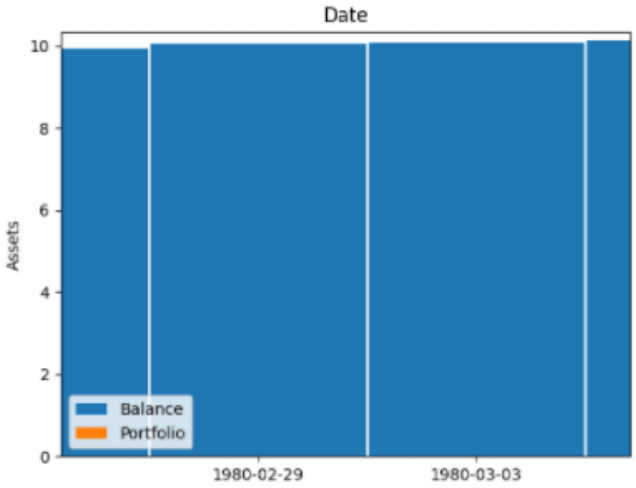
Success:

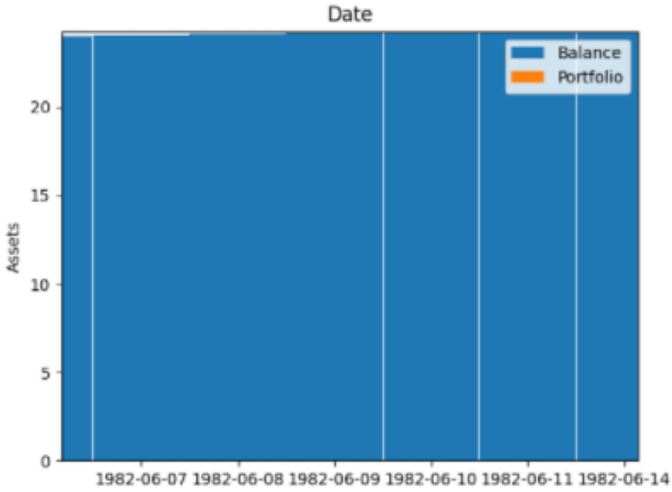
After 1000 transaction(s), your profit is 24.54233 dollars.

Στο παράδειγμα αυτό εμφανίζουμε και το αντίστοιχο ζητούμενο γράφημα, κάνοντας μεγέθυνση σε τυχαίες ημερομηνίες. Πρόκειται για “stacked bar chart”, που απεικονίζει σε κάθε χρονική στιγμή το balance που έχουμε και τον αριθμό σε assets του portfolio μας, όπως ακριβώς και πριν.









Όπως σχολιάστηκε και προηγουμένως, στις παραπάνω εικόνες παρατηρούμε ότι η κράτηση μίας μετοχής είναι αρκετά μικρή, καθώς η λογική μας βασίζεται σε ημερήσιες συναλλαγές. Συνεπώς δεν έχουμε μεγάλο όγκο στο portfolio. Για την κατασκευή του γραφήματος χρησιμοποιήθηκε η βιβλιοθήκη pyplot.