JOHANNES GUTENBERG UNIVERSITÄT MAINZ

JG|U

Institut für Informatik

Michael Wand
Christian Ali Mehmeti-Göpel

Wintersemester 2024/25

Data Science 3

Übung 9

Einführung in die Programmierung

Über dieses Übungsblatt

Wir haben bis jetzt gelernt Daten einzulesen, anzuzeigen und zu interpolieren. Nun möchten wir in unseren Daten Muster erkennen.

Aufgabe Geschachtelte Schleifen

Letzte Änderung: 06. July 2023, 13:26 Uhr
15 Punkte – im Detail

Ansicht: |

In dieser Aufgabe möchten wir an dem konzeptionellen Schritt arbeiten, unser Programm zunächst besser zu verstehen, bevor wir losprogrammieren. Sie haben seit Blatt3 sicherlich bemerkt, dass es manchmal notwendig ist, mehrere for Schleifen in einander zu schachteln, in etwa so :

```
# einfache for-schleife
for i in range(100):
        ...

# zweifach geschachtelte for-schleife
for i in range(100):
        for j in range(100):
                ...

# dreifach geschachtelte for-schleife
for i in range(100):
        for j in range(100):
                for k in range(100):
                ...
```

Dies ist eng verwand mit dem Begriff der Laufzeit, den wir später in diesem Semester tangieren werden, aber auch ausführlich in der Vorlesung Komplexitätstheorie kennenlernen werden. In dieser Aufgabe werden wir allerdings keine vollständige Laufzeitanalyse durchführen, wir wollen lediglich wissen **wie viele geschachtelte for Schleifen** unser Programm in der **naiven Ausführung** (ohne komplexe Algorithmen, die wir noch nicht kennen) benötigt. *Geben Sie für die folgenden Problemstellungen lediglich an, wieviele for Schleifen mindestens geschachtelt werden müssen, um das Problem zu lösen. Begründen Sie ihre Antwort kurz (1-2 Sätze maximum).*

1) Gegeben ein Array `arr` und eine Zahl `num`, sind alle Zahlen in `arr` größer als die Zahl `num`?

2) Gegeben zwei quadratischen Matritzen `mat1` und `mat2` gleicher Größe, berechnen Sie das Matrixprodukt beider Matritzen.

3) Gegeben eine quadratische Matrix `mat`. Geben Sie die Summe aller Einträge auf der Diagonale aus.

Wir möchten eine 10-Stellige Pin erraten, die nur aus den Ziffern 0-9 besteht. Wir haben dazu eine Funktion `check_pin(pin)` zur Verfügung, der wir eine `pin` (als String, sonst haben wir Probleme mit führenden Nullen) geben können und die uns entweder `True` oder

`False` zurückgibt, je nach dem ob die Pin stimmt oder nicht.

4a) Wir möchten bei einer vorgegebenen Liste von 100 Passwörtern überprüfen, ob eines passt.

4b) Wir möchten alle möglichen Passwörter durchprobieren.

5) Gegeben drei Listen: eine mit Schuhen `shoes=["vans", "boots", ...]`, Tshirts `shirts=["dotted red shirt", "elegant black shirt", ...]` und Hosen `pants=["jeans", "jogging pants", ...]`. Geben Sie alle möglichen Kombinationen von Hosen, Tshirts und Schuhen aus, die mit den Kleidungsstücken kombinierbar sind.

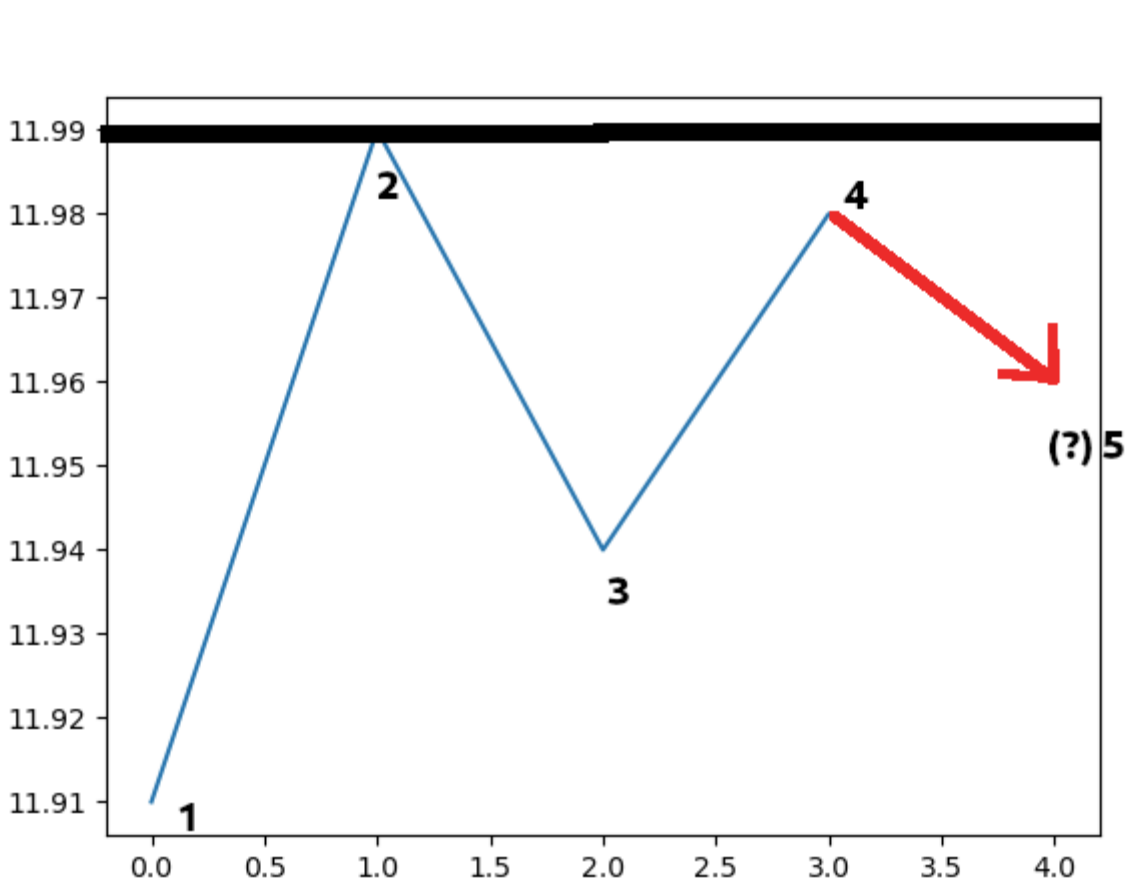
6) Suchen Sie sich zwei Aufgaben aus den Aufgabenteilen 1-5 aus (4a und 4b zählen als zwei Aufgaben) und programmieren Sie diese nach. **Verwenden Sie nur reine `for` Schleifen (oder ggf. auch rekursive Funktionen) und keine vorgefertigten Funktionen oder externe Bibliotheken.** Verwenden Sie zum Testen folgende Werte, Zahlen, Matritzen, Funktionen und Listen:

```
zahl = 80
arr = [1,100,26,-10,50,90,30]
mat1 = [[1,2,5],[2,7,1],[1,7,7]]
mat2 = [[5,1,8],[2,2,2],[1,9,9]]

def check_pin(pin: str):
    """Checks if a pin is correct. Returns a bool.
        Takes a pin given as a string as input.
    """

        assert isinstance(pin, str), "Pin needs to be a string"
        assert len(pin) == 10, "Pin needs exactly 10 digits"
        return pin == "1234543210"

shoe=["vans", "boots", "chucks", "heels"]
shirts=["dotted red shirt", "elegant black shirt", "trashy pink shirt"]
pants=["jeans", "jogging pants", "yoga pants", "shorts"]`
```

Aufgabe Trading Patterns

Letzte Änderung: 06. July 2023, 13:26 Uhr
25 Punkte – im Detail

Ansicht: |



Das "double top" Trading Pattern. Punkte 2 und 4 sind ungefähr auf der gleichen Höhe und bilden das "Doppelte Hoch". Das Muster sagt (angeblich) einen Abfall des Kurses vorraus.

Wenn es um das Handeln an der Börse (oder heutzutage sogar auf der Blockchain) geht, tobt schon sehr lange die Diskussion ob es prinzipiell möglich ist, Marktbewegungen anhand von Mustern vorherzusagen. Viele Trader behaupten, anhand von Mustern (wie in der Illustration zu sehen) Börsenbewegungen vorraussagen zu können. Ein Berühmter Kritiker solcher Praktiken behauptet:

"Ein Affe mit verbundenen Augen, der Pfeile auf die Finanzseiten einer Zeitung wirft, könnte ein Portfolio auswählen, das genauso gut abschneidet wie eines, das von Experten sorgfältig ausgewählt wurde."

Da wir nun Data Scientists sind, sind wir in der lage solche Behauptungen einfach zu überprüfen. Wir wollen anhand des Beispiels des in der Illustration gezeigten "double top" Musters eine solche Behauptung auf einem echten Datensatz überprüfen. Als Datensatz verwenden wir historische daily Ethereum Preise im Zeitraum 2015-2018, die in LMS beiliegen.

Das "double top" Pattern zeichnen sich folgendermaßen aus: erst steigt der Preis (1 -> 2), dann fällt der Preis (2 -> 3), dann steigt der Preis erneut auf (fast) das gleiche Niveau wie bei 2 (3->4). Wenn dieses Muster erfüllt ist, soll der Preis im Anschluss angeblich meistens fallen (4->5).

1a) Lesen Sie die Daten aus der .csv-Datei ein mit dem folgenden Befehl ein; so können Sie auch die Datumsspalte korrekt extrahieren:

```
import numpy as np
from datetime import datetime


str2date = lambda x: datetime.strptime(x, '%Y-%m-%d')
data = np.genfromtxt("ethereum.csv", delimiter=',', dtype=None, converters = {0: str2date}, encoding='utf-8')
dates = [d[0] for d in data]
prices = [d[5] for d in data]
```

und extrahieren Sie die Spalte, die dem Preis entspricht (`price(USD)`).

1b) Schreiben Sie eine Funktion `is_double_top(arr_part)`, die eine Teilsequenz der Länge 4 (also einfach ein Array mit 4 Zahlen) unserer Preisdaten erhält und `True` zurückgibt, falls die Sequenz einem "double top" Pattern entspricht, ansonsten `False`. Eine Sequenz der Länge 4 erfüllt das "double top" Pattern genau dann, wenn die oben in rot genannten Bedingungen erfüllt sind.

Hinweis: Die Werte an Datenpunkt 2 und 4 müssen nur ungefähr übereinstimmen. Wir setzen dazu eine Toleranz von 0.01; d.h. wir sagen dass die Werte d_2 und d_4 approximativ gleich sind, wenn $|d_2 - d_4| < 0.01$. Verwenden Sie `abs(a-b)`, um zu sehen, wie weit die Werte `a` und `b` auseinanderliegen.

1c) Wir möchten nun über die komplette Sequenz der eingelesenen Preisdaten iterieren und feststellen, wann das "double top" Pattern vorkommt. Dazu schieben wir ein "Fenster" der Länge 4 über die komplette Sequenz und überprüfen für jeden Ausschnitt, ob das Pattern vorkommt. Konkret werden also erst die Datenpunkte 1-4 auf das Pattern ürberprüft, dann die Datenpunkte 2-5, 3-6 etc. Immer wenn ein "double top" Pattern gefunden wird, soll auf der Konsole die Folgende Ausgabe erscheinen: `double top pattern found: [datum]`, wobei `[datum]` das Datum des Tages ist, an dem das Pattern beginnt.

Hinweis: verwenden Sie die Slicing Notation. `arr[10:20]` schneidet die Werte mit Index 10-19 aus dem Array `arr` aus.

Der Preis soll angeblich oft fallen, nachdem das Pattern auftritt; wir möchten das überprüfen.

2a) Schreiben Sie eine Funktion `check_prediction(arr_part)`, die eine Teilsequenz der Länge 5 unserer Preisdaten erhält und `True` zurückgibt, falls die Vorhersage des Patterns (der Preis fällt von Datenpunkt 4 zu 5) korrekt ist, ansonsten `False`.

2b) Ergänzen Sie ihre `for`-Schleife so, dass wenn ein "double top" Pattern detektiert wird, auch überprüft wird ob die Vorhersage stimmt. Zählen Sie wieviele der Vorhersagen tatsächlich gestimmt haben. Geben Sie aus, wieviele Vorhersagen es insgesamt gab und wieviel Prozent davon korrekt war.

Hinweis Sie können dazu beispielsweise eine (anfänglich leere) Liste `pred` verwenden, der `True` bei jeder richtigen Vorhersage und `False` bei jeder falschen Vorhersage angehängt wird. Sie können dann am Ende einfach den Mittelwert mit `np.mean(pred)` bilden, der die gewünschte Wahrscheinlichkeit berechnet.

2c) Welche Einschränkungen hat ihr Ergebniß? Denken Sie dass ihr Resultat sich zukünftigen ETH Daten oder andere Daten verallgemeinern lässt?