Semesterarbeit Teil 2 (Beat Kappert)

plot_single.py

Funktionsgraph:

```
y = x * sin(1/x)
```

Um eine komplexere Funktion auf das ganze NumPy Array `x` anzuwenden, wird diese separat definiert (`waveFunction`) und dann mit `npy.vectorize` angewendet.

plot_multiple.py

Mehrere Funktionsgraphen in der selben Graphik:

- e^x
- ln(x)

Dazu die erste Winkelhalbierende y = x, weil man den Graph einer Umkehrfunktion durch Spiegelung an dieser Geraden bekommt.

Speziell: Für den natürlichen Logarithmus muss die x-Achse auf positive Werte beschränkt werden (sonst Warnung von NumPy). Deshalb sind auch zwei separate `plot` Befehle nötig.

pie_bar.py

Lädt die CSV Datei `programming_language_popularity.csv` (zur Abwechslung mit `npy.loadtxt`) und stellt die Daten im gleichen Fenster nebeneinander als Torten- und Balkendiagramm dar. (Das Programm muss im Verzeichnis ausgeführt werden, wo sich die CSV Datei befindet.)

Es geht um den prozentualen Anteil für Programmiersprachen bei Google Suche (Google Trends Daten). Da die Datei nicht 100% liefert, wird der fehlende Anteil als "übrige" ergänzt. So bleibt das Programm etwas flexibler.

Weil der Name der Programmiersprache C# in der CSV Datei steht, darf das Kommentarzeichen nicht Hashtag sein (Parameter `comments` von `npy.loadtxt` wechselt auf ein selteneres Zeichen).

histogram_stock_price_change.py

Hier geht es um die statistische Analyse der Änderung zwischen Tagesschlusskursen (Close) an der Börse von einem Tag zum nächsten.

Dieses Programm öffnet einen interaktiven Dialog, mit dem ein CSV File ausgesucht werden kann. Zur Demonstration liegt `DAX_DAILY_1996_2018.csv` bei. Das sind die Tageskursdaten des DAX (Deutscher Aktienindex), so wie sie bei Yahoo Finance heruntergeladen werden können. [Weil es auf Mac OS einen Bug mit dem GUI für die Fileauswahl gibt, erkennt das Programm mit `platform.system()` das Betriebssystem und lädt unter Mac OS das File `DAX_DAILY_1996_2018.csv` direkt.]

Aus den Daten wird der bereinigte Tages-Schlusskurs (adjusted close) in das NumPy Array `close_history` geladen. Daraus wird das Array `daily_price_changes` mit allen Kursänderungen abgeleitet. Als Kennzahl für eine Kursänderung wird nicht einfach die Differenz zweier Tageskurse verwendet, denn wenn der DAX früher um 100 Punkte von 3900 auf 4000 stieg, war das spektakulärer als wenn er heute von 11900 auf 12000 steigt. Ein Prozentwert ist auch nicht praktisch, denn nach einem Anstieg um 1% kommt man mit einem Rückgang um 1% nicht wieder zum Ausgangspunkt. Stattdessen wird als Kennzahl für die Kursänderungen der natürliche Logarithmus des Quotienten aus zwei Kursen verwendet:

```
Kursänderung = ln(Vortageskurs / Tageskurs)
```

Daraus resultiert eine Kennzahl für jede Kursänderung, die ca. im Bereich -0.08 bis +0.08 liegt. Diese Kennzahlen erlauben einen historischen Vergleich von Kursschwankungen, nach oben und nach unten.

Im Histogramm erkennt man, dass die Kennzahlen für Kursänderungen selten einen Betrag von 0.02 überschreiten. Angewendet auf einen aktuellen DAX Stand von 12000 Punkten könnte man ausrechnen, wie gross der Rückgang für Kennzahl 0.02 wäre:

```
12000 / e^0.02 ≈ 11762
```

Ein Börsenhändler darf also damit rechnen, dass ein Kursabschlag um 238 Punkte (von 12000 auf 11762) oder mehr über Nacht doch selten sein müsste. Diese Information ist vielleicht nützlich für die Abschätzung von Risiko und das Setzen von Stop Loss Limiten zur Absicherung.

Das Histogramm zeigt auch, dass die Tage mit Kurswachstum überwiegen. Es ist also einfacher, mit Long Positionen Geld zu verdienen als mit Shorts.

polar_circles.py

Ein Beispiel aus dem Internet, auf welches ich zufällig gestossen bin. Ich fand es interessant, dass man mit der Sinus-Funktion in Polarkoordinaten so einfach zwei Kreise zeichnen kann. Durch Erweiterung mit dem Cosinus sind es nun vier Kreise und für mich eine neue Sichtweise auf diese beiden trigonometrischen Funktionen.