

Serie 10

Aufgabe 1:

Fügen Sie zur Klasse `Polynom` aus den vorherigen Übungen eine Methode `plot(self, x_1, x_2)` zum Plotten der Funktion in einem beliebigen Intervall $[x_1, x_2]$. Beschriften Sie die Achsen sinnvoll und geben Sie dem Plot einen Titel.

Aufgabe 2:

Lesen Sie die Datei `data.txt` ein und erstellen Sie eine Liste aller Zeichen die in der Datei vorkommen (ohne Duplikate). Erstellen Sie eine zweite Liste, die die Häufigkeit jedes Zeichens in der Datei angibt (in der gleichen Reihenfolge wie die erste Liste). Erstellen Sie eine Balkendiagramm, das die Häufigkeit der Zeichen darstellt. Beschriften Sie die Achsen sinnvoll und geben Sie dem Plot einen Titel. Formattieren Sie das Bild so, dass die Zeichen auf der x-Achse lesbar sind (Breite des Bildes anpassen).

Aufgabe 3:

Visualisieren Sie den Newton-Algorithmus aus Serie 8, Aufgabe 2. Schreiben Sie dafür eine Funktion, die in jedem Schritt des Newton Algorithmus die Funktion im Intervall $[x_k - 1.5 * |x_0 - x_1|, x_k + 1.5 * |x_0 - x_1|]$ plottet. Markieren Sie $(x_k, f(x_k))$. Zusätzlich soll die Approximationsgerade, deren Nullstelle der Wert x_{k+1} ist, geplottet werden. (Also die Gerade durch $(x_k, f(x_k))$ mit Steigung $f'(x_k)$). Der Schnittpunkt mit der x_Achse (x_{k+1}) soll mit einem roten Punkt markiert werden. Formatieren Sie die Grafik sinnvoll, mit ausagekräftiger Legende und Achsenbeschriftungen.

```
In [ ]: def newton(f, df, x0, tol=1e-10, max_iter=100):
    """
    f : function
        The function for which we want to find a root.
    df : function
        The derivative of the function f.
    x0 : float
        The initial guess for the root.
    tol : float
        The tolerance for convergence.
    max_iter : int
        The maximum number of iterations.

    Returns:
    float
        The estimated root of the function f.
    """
    x = x0
    for i in range(max_iter):
```

```

fx = f(x)
dfx = df(x)
if abs(fx) < tol:
    return x
if dfx == 0:
    raise ValueError("Derivative is zero. No solution found.")
x -= fx / dfx
raise ValueError("Maximum iterations exceeded. No solution found.")

```

Aufgabe 4:

Visualisieren Sie die Konvergenzgeschwindigkeit des Bisektionsverfahrens aus der Vorlesung und vergleichen Sie es mit dem Newtonverfahren von oben. Suchen Sie dafür mit beiden Methoden die Nullstelle der Funktion $f(x) = x^3 - x - 2$ im Intervall $[1, 3]$ (für das Bisektionsverfahren) bzw. mit Startwert $x_0 = 2$ (für das Newtonverfahren). Erzeugen Sie dafür eine Folge von Approximationen der Nullstelle x_0, \dots, x_n (die Schritte des Bisektions/Newton Verfahren) und plotten Sie die Werte $(i, |f(x_i)|), i = 0, \dots, n$ mithilfe von `matplotlib` für beide Verfahren im selben Plot. Verwenden Sie sowohl normal skalierte als auch logarithmisch skalierte y-Achsen. Vergleichen und interpretieren Sie die Plots für das Bisektionsverfahren und das Newtonverfahren. Finden Sie selbst noch eine weitere Funktion, bei der beide Verfahren konvergieren, und vergleichen Sie die Konvergenzgeschwindigkeit erneut.

Aufgabe 5:

Schreiben Sie eine Funktion, die ein Spielfeld für **Tic/Tac/Toe** visualisiert. Die Funktion soll das Spielfeld als Matrix übergeben bekommen mit Einträgen `" "`, `"x"`, `"o"`. Plotten Sie zuerst das Spielfeld und fügen Sie danach Kreise für `"o"` und Rechtecke für `"x"` (oder Formen Ihrer Wahl) hinzu.

Aufgabe 6:

Passen Sie den Code der Klasse `TextGenerator` aus dem Kapitel "Textgenerierung mit Markov-Ketten" aus den Vorlesungsnotizen so an, dass Sie nicht das wahrscheinlichste nächste Wort auswählen, sondern das wahrscheinlichste nächste Zeichen. Ihr Wörterbuch soll also für alle aufeinanderfolgende Zeichen `(c1, c2)` des Texts eine Liste aller möglichen nächsten Zeichen `c3` die im Text vorkommen speichern. Verwenden Sie dieses angepasste Programm, um aus der Datei `data.txt` einen neuen Text zu generieren. Das Programm funktioniert, wenn etwas rauskommt das auf den ersten Blick wie Englische Wörter aussieht, z.b.

*A peed dold!) As XP) be re ine-can ist may Pythe nothosion on
thoullodectimeramicturrection fack synamix staing in agent tat() is all thons usement,
i.e.[165] tor be expere...*