Datentypen und Fallunterscheidungen

A) Datentypen

B) Text-Eingaben und -Ausgaben

C) Boolsche Ausdrücke und Fallunterscheidungen

D) Hausaufgaben

A) Datentypen

Die wichtigsten Datentypen in Matlab haben Sie bereits kennengelernt: Zahlen, Vektoren und Matrizen. Zahlen sind in Matlab normalerweise definiert als Fliesskommazahlen mit doppelter Präzision (double). Es gibt aber z.B. auch ganze Zahlen (int), Wahrheitswerte (logical), die nur die Werte 0 oder 1 annehmen können, und Buchstaben (char). Welchen Datentyp man wählt, hängt davon ab, welche Art von Daten man damit darstellen möchte.

Matrizen und Vektoren sind standardmäßig aus Zahlen vom Typ double zusammengesetzt. Es gibt aber auch Matrizen und Vektoren, die aus ganzen Zahlen, Wahrheitswerten oder Buchstaben bestehen. Allerdings kann man auf diese nicht unbedingt alle Operationen anwenden, die man für Matrizen aus "normalen" Zahlen zur Verfügung hat.

Zur Umwandlung des Typs einer Variablen benutzt man den Namen des gewünschten Datentyps. z.B. macht logical(a) aus der Variable a eines beliebigen Datentyps einen Wahrheitswert.

Fließkommazahlen können sehr viele Stellen hinter dem Komma haben, z.B. die Zahl PI. Wenn Sie sich pi anzeigen lassen, sehen Sie nur vier Nachkommastellen. Tatsächlich rechnet Matlab aber mit sehr viel mehr Stellen. Trotzdem muss man sich dessen bewusst sein, dass Computer grundsätzlich nur mit endlicher Genauigkeit (festgelegt durch die Konstante eps) rechnen können, also die Nachkommastellen an einer bestimmten Stelle abgeschnitten werden. Auch gibt es eine Obergrenze des bearbeitbaren Zahlenbereichs (Konstante realmax). Oberhalb dieser Zahl gibt es für Matlab nur noch eine Zahl, mit der zuverlässig gerechnet werden kann: inf (infinity, also Unendlich).

Ganze Zahlen haben ebenfalls einen beschränkten Darstellungsbereich. Je nach dem, wie groß die Zahlen sind, die man mit einer ganzzahlingen Variablen darstellen möchte, gibt es in Matlab verschiedene Typen, die Unterschiedlich viel Speicherplatz brauchen: int8, int16, int32 und int64. Dabei gibt die Zahl jeweils an, wie viele bit an Speicherplatz verwendet werden, z.B. können mit int8 28=256 Zahlen dargestellt werden. Da sowohl positive als auch negative Zahlen und Null im Bereich von int8 enthalten sind reicht der darstellbare Bereich von -128 bis 127. Wenn man sicher ist, dass man nur positive Zahlen verwenden möchte, kann man auch uint8, uint16, uint32 und uint64 ("unsigned integer") verwenden. Für uint8 geht der Zahlenbereich von 0 bis 255.

T3_A1) Man kann mit dem Befehl format umstellen wie viele Nachkommastellen einer Fliesskommazahl gezeigt werden. Probieren Sie aus, wie pi und wie 100*pi angezeigt werden nachdem sie angegeben haben: format long, format long e, format short e, format short
Für welche Anwendungen ist welches Format am besten geeignet?

T3_A2) Computer können nur einen begrenzten (wenn auch ziemlich großen) Zahlenbereich darstellen und verarbeiten. Schauen Sie sich die größte und die kleinste positive Zahl an, mit der Ihr Rechner mit Matlab zuverlässig rechnen kann: realmax, eps
Was passiert, wenn Sie diesen Bereich über- bzw. unterschreiten? Probieren Sie z.B. realmax*10, eps/2, realmax+1, 10-eps, 10-eps

T3_A3) Zwei besondere Konstanten in Matlab freuen einen meistens nicht, wenn man sie als Ergebnis einer Berechnung geliefert bekommt: inf und nan. Schauen Sie in der Hilfe nach, was diese bedeuten. Was ist der Unterschied zwischen ihnen? Versuchen Sie, die beiden Konstanten durch Rechenoperationen zu erzeugen. Erzeugen Sie zwei Matrizen, in denen einmal an einer Stelle nan und in der anderen an einer Stelle inf vorkommt, während alle anderen Matrixelemente normale Zahlen sind. Versuchen Sie mit beiden Matrizen zu rechnen. Was passiert?

T3_A4) Es gibt Bereiche, bei denen gebrochene Werte keinen Sinn ergeben, sondern ausschließlich ganze Zahlen (integer), z.B. wenn sie ein Element eines Vektors indizieren wollen. Stellen Sie sich vor, Sie haben einen Vektor v, der 100 Datenpunkte einer Messreihe enthält und wollen wissen, wie groß Ihr Messergebnis nach 1/3 der Zeit war. Probieren Sie aus:
v=0.1:0.1:10; l=length(v); ind=l/3; m=v(ind)
(Wie sieht die Antwort von Matlab im Vergleich aus wenn Sie den Messpunkt nach 1/2 der Zeit betrachten wollen?)
Um die Fehlermeldung los zu werden, können Sie den Index explizit zu einer Integer-Variable machen:
v=0.1:0.1:10; l=length(v); ind=int16(l/3); m=v(ind)
Eine Alternative dazu ist, weiterhin mit dem Standard-Datentyp double zu arbeiten, aber die entsprechende Zahl zu runden:
v=0.1:0.1:10; l=length(v); ind_a=round(l/3); m_a=v(ind_a)
Allerdings kann man so eine Rundung leicht vergessen, insbesondere wenn mit den Zahlen weiter gerechnet wird. Sicherer ist deshalb, integer zu verwenden, wenn gebrochene Zahlen keinen Sinn machen. Probieren Sie aus:
ind=int16(l/3); m=v(ind); ind2=ind/2; m2=v(ind2) versus
ind_a=round(l/3); m=v(ind_a); ind2_a=ind_a/2; m2_a=v(ind2_a)
Welchen Datentyp hat ind2? Welchen ind2_ a? Was sind ihre Werte?

T3_A5) Ein weiterer wichtiger Datentyp sind Wahrheitswerte (logical). Diese können nur zwei Werte annehmen "wahr", symbolisiert durch 1, und "falsch", symbolisiert durch 0. Logical-Variablen können durch den Befehl logical generiert werden, z.B. a=logical(0). Welchen Wert und welchen Typ hat a im Workspace?
Definieren Sie außerdem b=logical(1). Was passiert bei c=logical(2)?
Den Umgang mit Wahrheitswerten üben wir heute in Teil C)

T3_A6) Wenn man mit Matlab Texte verarbeiten möchte, werden diese als Zeichenketten (array of char) dargestellt. Die Syntax um eine Zeichenkette zu generieren und einer Variablen zuzuweisen ist:
text='Texte werden in Hochkommata eingeschlossen'
Mit solchen Zeichenketten kann man in gleicher Weise arbeiten, wie mit Vektoren, um einzelne Zeichen oder Zeichenfolgen zu indizieren oder Zeichenketten hintereinanderzuhängen. Leerzeichen zählen dabei genauso mit wie alle anderen Zeichen. Probieren Sie aus:
text(1), teil1=text(1:6), teil2=text(17:25), syntax=[teil2 ' ' teil1 ' ' teil2] (damit man den letzten Ausdruck besser lesen kann hier noch mal mit _ für jedes Leerzeichen: syntax=[teil2_'_'_teil1_'_'_teil2])

B) Text-Eingaben und -Ausgaben

Die meisten Programme müssen in irgendeiner Weise mit dem Benutzer interagieren. Die modernste Möglichkeit dafür sind grafische Benutzeroberflächen (GUIs, graphical user interfaces), die man in Matlab relativ komfortabel gestalten kann. Mit diesen werden wir uns im Rahmen des Kurses allerdings nicht beschäftigen. Wir beschränken uns auf die "herkömmliche" Form der Bildschirmausgaben als Text und Abbildungen und der Benutzereingaben per Tastatur im Befehlsfenster.

T3_B1) Wir wissen bereits, dass Berechnungsergebnisse standardmäßig im Command Window ausgegeben werden außer sie werden durch ein ";" am Ende der Zeile unterdrückt.
Es kann jedoch sinnvoll sein, dem Benutzer auch andere Informationen per Textausgabe zur Verfügung zu stellen. Dies geschieht durch die Funktion disp, der eine Zeichenkette übergeben wird.
Einfach weil es zu einem Programmierkurs dazugehört, erzeugen Sie ein Skript hello_world.m das als einzige Zeile enthält
disp('hello world')
Weil das doch ein wenig langweilig ist, ergänzen Sie hello_world so, dass es Ihnen außerdem in freundlicher Art noch das aktuelle Datum verrät, verwenden Sie dafür die Funktion date.

T3_B2) Mit einer speziellen Art der Textausgabe sind Sie inzwischen bestens vertraut: den rot geschriebenen Fehlermeldungen. Diese können Sie auch selber erzeugen durch den Befehl error, z.B. mit error('so geht es ja nun nicht!')
Die Funktion error macht jedoch noch mehr als eine rote Textausgabe, sie bricht außerdem die Ausführung eines Programms ab. Probieren Sie das aus, indem Sie ein den error-Befehl an verschiedenen Stellen eines längeren Skripts oder Funktion einfügen (z.B. kompliziert.m von gestern). Wenn Sie in der Datei keine Semikolon verwenden oder den Editor benutzen, können Sie nachvollziehen, welche Schritte jeweils ausgeführt werden.

T3_B3) Wenn man den Benutzer warnen will, dass gerade wahrscheinlich etwas schief läuft, aber nicht das ganze Programm abgebrochen werden sollte, gibt es dafür den Befehl warning. Probieren Sie auch diesen aus.

T3_B4) Man kann Textausgaben auch mit der Ausgabe von Variablenwerten mischen. Der Befehl dafür ist sprintf. Dieses ist eine Funktion, die als erstes Argument eine Zeichenkette übergeben bekommt, die ausgegeben werden soll, bei der an Stelle der Variablenwerte die Schlüsselbegriffe %g (bei Variablen vom Typ double, int, oder logical) oder %s (bei strings) verwendet werden. Als weitere Argumente werden die Variablennamen in der Reihenfolge übergeben, in der sie im Text auftauchen.
Probieren Sie aus: a=0.2; t='text'; sprintf('jetzt koennen wir Zahlen wie z.B. %g und auch %s darstellen.',a,t)
Erweitern Sie eins Ihrer Spaghettiskripte oder Funktionen von gestern so, dass die Ausgabe der Länge mit Angabe der Einheit erfolgt.
*) Für Leute mit zu viel Zeit oder einem Hang zu schönen Bildschirmausgaben: mit sprintf kann man die Formatierung sehr exakt festlegen (z.B. Anzahl Nachkommastellen) - wie das geht steht in der Hilfe. Dort gibt es auch Hinweise zum entsprechenden Befehl fprintf mit dem man formatiert in Dateien schreibt.

T3_B5) Zeichenketten werden auch gebraucht, um Abbildungen zu beschriften. plotten sie für x1=-10:0.1:10 den Vektor y1=x1^2. Benutzen Sie die Befehle title, xlabel, ylabel und legend, um Ihre Grafik zu beschriften. title, xlabel, ylabel bekommen jeweils eine Zeichenkette als Eingabeargument, legend so viele durch Komma getrennte Zeichenketten wie Vektoren in der Grafik dargestellt sind.
Probieren Sie aus mehrer Kurven in eine Abbildung zu plotten. Hierfür erzeugen Sie für einen zweiten x-Vektor x2=-5:0.1:5 die y-Werte y2=2*x2^2. Um beide Kurven in eine Abbildung zu plotten, muss man jeweils die x- und y-Werte explizit angeben, also in unserem Fall plot(x1,y1,x2,y2). Beschriften Sie in der Legende beide Kurven.

T3_B6) Häufig möchte man dem Benutzer nicht nur etwas mitteilen, sondern auch Benutzereingaben erhalten. Einen Spezialfall davon haben Sie bereits kennengelernt, den Befehl pause. Wenn auch der Wert einer Eingabe abgefragt werden soll, verwendet man den Befehl input. a=input('Eingabe') weist der Variablen a eine Zahl, oder einen Variablenwert zu. Probieren Sie aus:
a=input('Wert für a: ') -> bei Abfrage 7 eingeben
b=input('Wert für b: ') -> bei Abfrage a eingeben
c=input('Wert für c: ') -> bei Abfrage xyz eingeben

Wenn Text eingelesen werden soll, braucht der Befehl als zweiten Eingangswert die Option 's':
c=input('Wert für c: ','s') -> bei Abfrage xyz eingeben
d=input('Wert für d: ','s') -> bei Abfrage 7 eingeben
e=input('Wert für e: ','s') -> bei Abfrage b eingeben

T3_B7) Man kann Zeichenketten in Zahlen umwandeln und umgekehrt. Die Befehle dafür sind num2str (number to string) und str2num. Probieren Sie aus:
a=1.1, a_char=num2str(a), a_neu=str2num(a_char)
Was passiert bei a_neu=str2num(a)? Warum?
b='xyz', b_num=str2num(b)
Diese Eigenschaft ist besonders praktisch, um Zeichenketten automatisch zu generieren, z.B. für die systematische Benennung von Dateien.
Schreiben Sie ein Programm das den Benutzer zuerst nach einer Zeichenkette fragt, dann nach einer Zahl. Speichern Sie die Zeichenkette in einer Datei deren Namen

Sie aus 'datei' und der eingegebenen Zahl generieren, z.B. wenn der Benutzer 5 eingegeben hat, nennen Sie die datei 'datei5.mat'.
Achtung, wenn der Dateiname in der Variablen dateiname gespeichert ist, funktioniert save dateiname variablenname nicht. Hierfür braucht man die kompliziertere (und leider sehr wenig intuitive) Syntax
save(‘dateiname’,'variablenname').

T3_B8) Achtung! Matlab hindert Sie nicht daran, mit Zeichenketten zu rechnen! Probieren Sie aus:
a='1', a2=2*a, a2_ok=2*str2num(a),
b='xyz', b+1

T3_B9) Um sich Programmtexte im command window anzeigen zu lassen, gibt es den Befehl type. Probieren Sie in ihrem Arbeitsverzeichnis aus type spaghettiskript. Praktischerweise gilt type nicht nur für Ihre eigenen Programme, sondern auch für die in Matlab enthaltenen, so dass man sich diese ohne lange Suche ansehen und sich etwas abgucken kann. Probieren Sie z.B. type imagesc (nein, den Programmcode müssen Sie noch nicht verstehen). Bei manchen Kern-Funktionen lässt Matlab sich allerdings nicht auf die Finger schauen, z.B. type sin

C) Boolsche Ausdrücke und Fallunterscheidungen

Bisher entsprachen alle Programme im Kurs einem einfachen Kochrezept: Einzelne Schritte wurden der Reihe nach abgearbeitet. Jetzt gehen wir einen Schritt weiter und führen Alternativen ein, wie an einer bestimmten Stelle der weitere Programmablauf fortgesetzt werden soll. (Z.B. "Fügen Sie jetzt dem Muffinteig Heidelbeeren hinzu, alternativ können Sie auch Himbeeren verwenden, wenn diese Ihnen besser schmecken.")

T3_C1) Kehren wir noch mal zurück zu unserem Spaghetti-Beispiel: Normalerweise interessiert einen beim Konsum von Mensa-Spaghetti weniger ihre Länge als eher die Frage "sollte ich die wirklich nehmen?". Auch dabei kann uns Matlab weiterhelfen. Zur Beantwortung dieser Frage brauchen wir zunächst ein Kriterium. Nehmen wir an, ein besonders hungriger Student geht zur Mensa. Für ihn ist das Kriterium zunächst "Ist die Portion groß genug?". Allerdings müssen wir das noch etwas präzisieren: "groß genug" könnte heißen "wiegt die Portion mehr als 300g?". Der Student muss also zunächst einen Test durchführen (die Spaghetti wiegen) und dann sein Verhalten entsprechend davon abhängig machen. Für die Kaufentscheidung des Studenten gilt also der Algorithmus:

Wenn Gewicht>300g:
    nimm den Teller
Sonst:
    sag dem Thekenpersonal: "ich hätte gerne noch einen Nachschlag"

Übersetzt in Matlab-Syntax sieht das so aus:

```
if Gewicht>300
   disp('Teller nehmen')
else
```

```
    disp('ich hätte gerne noch einen Nachschlag')
end
```

Schreiben Sie ein Programm, das den Koch nach dem Gewicht der Spaghettiportion fragt und abhängig von seiner Eingabe entsprechend antwortet.

T3_C2) Bei der Auswertung der Bedingung in einer if-Abfrage wird intern eine boolsche Variable benutzt. Diese können Sie auch explizit anlegen und ansehen:

```
groß=(Gewicht>300)
if groß
   disp('Teller nehmen')
end
```

T3_ C3) Probieren Sie verschiedene Vergleichsoperatoren aus, indem Sie sich zwei Variablen mit Skalarwerten definieren und diese Vergleichen mit jedem der Vergleichsoperatoren:
<, >, ==, <=, >=, ~=
Wenden Sie die Vergleichsoperatoren auch auf zwei Matrizen gleicher Größe an. Welchen Typ hat die Ausgabe? Wie sieht die Rechenregel aus?

T3_C4) Vergleichsoperatoren auf Vektoren oder Matrizen anzuwenden, kann sehr hilfreich sein, um in Daten bestimmte Bedingungen zu finden. Laden Sie folgende Messreihe des pH Werts von Aquarienwasser: [phWerte.mat] (bzw pH-Werte_v6.mat)
Wie viele der Messpunkte sind Messfehler (zumindest wenn Ihre Fische normale Süßwasserfische sind und die Messreihe überlebt haben)? Wie häufig wurden zu große und wie häufig zu kleine Werte aufgezeichnet?

T3_C5) Manchmal möchte man zwei ganze Matrizen oder Vektoren aug Gleicheit testen und nicht ihre einzelnen Elemente. Dafür gibt es den Befehl isequal(M,N), der einen einzelnen Wahrheitswert zurückliefert. Dieser lässt sich nicht nur auf Zahlen, sondern auf alle Datentypen anwenden (wie wir später sehen werden auch auf selbst definierte). Insbesondere eignet er sich auch für Texte. Speziell für den Vergleich von strings gibt es außerdem den Befehl strcmp(t1,t2) (string comparison). Probieren Sie ==, isequal und strcmp für Variablen verschiedener Datentypen aus.

* T3_C6) Ein bisschen Logik: Erzeugen Sie sich Tabellen, in denen alle möglichen Kombinationen aus Belegungen von zwei logische Variablen verknüpft werden mit:
1. a & b (und)
2. a | b (oder)
3. xor(a,b) (exklusives oder)
4. ~a

* T3_C7) Mit den Befehlen

```
E=zeros(7); E(6,3:6)=ones(1,4); E(4,3:5)=ones(1,3); E(2,3:6)=ones(1,4);
E(2:6,2)=ones(5,1);
```

ergibt sich eine Matrix, welche mit der Funktion imagesc(E) einen Buchstaben zeigt. Erzeugen Sie davon durch Anwendung logischer Operatoren ein Negativbild.

T3_C8) Wenn man mehrere logische Variable verknüpft, kann es mehr als zwei Bedingungen geben. Diese werden mit

```
if (bedingung1)
   befehle1
elseif(bedingung2)
   befehle2
else
   befehle3
end
```
abgefragt. (Dabei kann es beliebig viele elseif-Teile geben).
Erweitern Sie das Programm aus T3_C1) so, dass der Student die Spaghetti nur kauft, wenn das Gewicht>300g ist und die Dichte>1.05, und dem Koch ansonsten eine der jeweiligen Situation angemessene Antwort gibt. Hierzu brauchen Sie als Schlüsselbegriff
elseif Bedingung (steht zwischen if und else, wird ansonsten genauso verwendet wie if)

*T3_C9) Nur für Leute mit zu viel Zeit oder gesteigertem Interesse an Programmierkonzepten:
Matlab sieht für Abfragen von vielen verschiedenen Fällen eine switch-Anweisung vor. Schauen Sie in der Hilfe nach, wie diese zu verwenden ist und schreiben Sie eine Funktion "arbeitsamt", die einen Namen (string) als Eingabe bekommt und diesen Namen dem zuständigen Sachbearbeiter zuweist und dessen Namen auf dem Bildschirm ausgibt. Zuständigkeit für Anfangsbuchstaben:
A-D: Frau Schmidt
E-H: Herr Klein
I-L: Frau Groß
M-P: Frau Mueller
Q-T: Herr Maier
U-Z: Herr Dimpfelmoser

D) Hausaufgaben

*T3_H1) Probieren Sie aus, welche Datentypen sich in welche umwandeln lassen. Erzeugen Sie sich je eine Variable der verschiedenen Ihnen bekannten Datentypen und wandeln sie jeweils den Typ in die anderen Datentypen. Was passiert z.B. wenn man aus Zeichen Zahlen macht und umgekehrt? Was passiert mit Vorzeichen und Nachkommastellen wenn man Umwandlungen in andere Typen vornimmt?

T3_H2) In folgender Datei sind die Merkmale der Katzen in einem Tierheim gespeichert: [katzen.mat].bzw [katzen_v6.mat] Im Vektor alter ist für jede Katze ihr Alter in Jahren gespeichert, im Vektor weibchen ein true, wenn es sich bei der Katze um ein Weibchen handelt, bei Katern ein false. Die Merkmale der Tiere sind jeweils in der gleichen Reihenfolge angeordnet.
Wie bekommt man heraus, wie viele Katzen jünger sind als 2 Jahre?
Wie die Anzahl der unter zweijährigen Weibchen?
T3_H3) Bei den Merkmale der Katzen im Tierheim [katzen.mat] findet sich auch eine Matrix farben. Diese besteht aus Wahrheitswerten und ist folgendermaßen aufgebaut:

Zeilen: 23 Katzen
Spalte 1: Katze hat schwarzes Fell
Spalte 2: Katze hat rotes Fell

Spalte 3: Katze hat weißes Fell
Spalte 4: Katze hat getigertes Fell

Wie viele Katzen haben getigertes Fell?
Wie viele Katzen haben rein getigertes Fell, ohne andere Farben?
Wie viele dreifarbige Glückskatzen (schwarz-weiß-rot) gibt es?
Wie viele einfarbige Katzen gibt es (wenn getigerte Katzen nicht als einfarbig angesehen werden)?

* oder **) T3_H4) (Wird später weiterverwendet) Schreiben Sie ein Programm, das Besuchern des Tierheims durch einen Dialog ermöglicht herauszufinden, ob es ihre Traumkatze bezüglich Alter, Geschlecht und Fellfarbe im Tierheim gibt. (Diese Aufgabe ist beliebig ausbaubar, Sie können sich selber überlegen, wie viele Details Sie in Ihrem Programm bedenken wollen. Wer gerne besonders schöne Lösungen überlegen möchte, kann dafür auch andere Aufgaben weglassen.)

T3_H5) Man kann den Typ einer Variablen im Programmtext testen (was z.B. nützlich ist wenn Benutzer eventuell absoluten Unsinn an eine Funktion übergeben könnten). Der Test, ob eine Variable einen Wahrheitswert beinhaltet heißt islogical. Finden Sie die anderen Abfragen heraus und schreiben Sie eine Funktion, die für eine beliebige Eingabe den Typ der Eingabe auf dem Bildschirm ausgibt.

T3_H6) Eine extrem wichtige Anwendung von if-Abfragen sind Tests, ob einer Funktion sinnvolle Parameter übergeben wurden. Ergänzen Sie Ihre gestern geschriebene spaghettifunktion aus T2_C5 so, dass bei unsinnigen Parameterübergaben eine Warnung ausgegeben wird.

*T3_H7) Erweitern Sie die Funktion diagonale.m vom letzten Kurstag (T2_H6) so dass eine Warnung erfolgt, wenn der Benutzer Eingabeparameter übergibt, die nicht der Spezifikation entsprechen.