2. IGOR – Objekte

2.1 Variable

Eine Variable in IGOR ist eine Zahl, die einen Namen trägt. So ist es zum Beispiel möglich, unter dem Namen *var1* die Zahl 4 abzuspeichern. Wichtig hierbei ist, dass IGOR Zahlen als Gleitkommazahlen (engl. Floating Point) speichert. Diese bestehen aus einer *Mantisse* und einem *Exponenten*, jedoch ist das Format der Darstellung für viele ungewöhnlich, daher ein Beispiel:

normal: $6.02 \cdot 10^{-23} = \text{Mantisse} \cdot 10^{\text{Exponent}}$

Schreibweise in IGOR: 6.02E-23

Weiterhin ist es wichtig, dass IGOR im Gegensatz zu uns bei 0 mit dem Zählen anfängt. Dies fällt schon bei der Betrachtung des Tabellenfensters auf, wo die erste Zeile mit 0 und nicht mit 1 deklariert ist (siehe Kapitel 1).
Um eine Variable zu benennen, nutzt man Befehl

```
variable var1 = 6.02E-23
```

mit diesem wird unter dem Namen *var1* die Zahl $6.02 \cdot 10^{-23}$ gespeichert. Jedoch gibt es auch hier einige Regeln für die Deklarierung von Variablen:

1. Es dürfen keine Ziffern zu Beginn stehen.
2. Der Name darf keine Leerzeichen enthalten, jedoch sind Unterstriche „_“ erlaubt.
3. Es dürfen **keine** Sonderzeichen benutzt werden.
4. Es darf keine Bindestrich/Minus „-„ verwendet werden.
5. Der Name darf Maximal 32 Zeichen lang sein.

Variablen können nach der Erstellung geändert werden; folgende Eingaben an der Kommandozeile zeigen dies:

```
variable var2 = 7
print 2*var2     (Igor gibt die Zahl 14 aus)
var2 = 3*var2    (var2 wird gleich 21 gesetzt)
print 2*var2     (Igor gibt jetzt die Zahl 42 aus)
```

Neben dem Datentyp `variable` gibt es noch den Datentyp `constant`, der eine unveränderliche, konstante Zahl enthält.
Die festgelegten Variablen gelten nur für das gerade geöffnete Experiment.
Um Zahlen (Beispiel: Naturkonstanten) universell für alle Experimente nutzen zu können, müssen diese mit der Deklaration `constant` (statt `variable`) als Text-Datei mit der Datei-Endung ".ipf" im *IGOR Procedures* Ordner abgelegt werden. Dieser Ordner ist ein Unterverzeichnis Ihres Igor Pro-Verzeichnisses. Die in der ".ipf"-Datei als `constant` abgelegten Größen werden beim Start von Igor aus der ".ipf"-Datei in das aktuelle Igor-Experiment eingelesen und können nicht irrtümlich verändert werden. In das *IGOR Procedures*-Verzeichnis sollte nur ein Link (eine Verknüpfung) auf die Originaldatei abgelegt werden, da diese sonst nach jedem Update erneuert werden muss.

2.2 String

Ein *String* in IGOR ist eine gespeicherte Zeichenkette, z.B. "abc" ist eine Zeichenkette (String) aus den Zeichen "a", "b", "c". Diese kann aus Ziffern oder Buchstaben bestehen. Da es in IGOR für jegliche Deklarierung eine Regel gibt, trifft dies auch hier zu. Hier können die gleichen Regeln bzgl. der Namensgebung wie für Variablen in Kapitel 2.1 angewendet werden. Jedoch wird der Vorgang durch den Befehl

```
string str1 ="pcprakt"
```

eingeleitet. Hierbei wird ein String mit dem Namen *str1* abgespeichert (wieder nur für das Experiment) der als Inhalt die Zeichenfolge *pcprakt* hat.
Die Deklaration

```
string str2 ="123"
```

erzeugt **nicht** die Zahl 123 (einhundertdreiundzwanzig), sondern die Ziffernfolge "123". Diese Ziffernfolge ist **keine** Zahl! Sie ist ein String wie jeder andere.
Strings in IGOR können *addiert* werden. Diese Addition ist von der Addition von Zahlen vollkommen verschieden, denn bei der Stringaddition werden die beiden Strings nur aneinander gehängt. Das subtrahieren ist **nicht** möglich. Hier ein Beispiel:

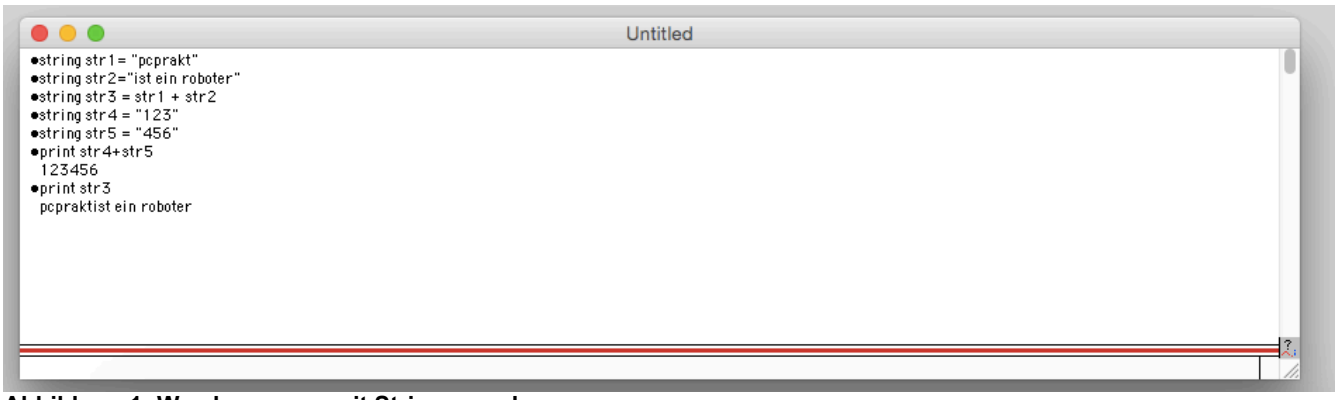

Abbildung 1: Was kann man mit Strings machen

Wie man in Abbildung 5 erkennen kann, wurden zuerst die Strings *str1-str5* definiert, wobei String *str3* die Addition von *str1* und *str2* ist. An den Printzeilen ist zu erkennen, dass IGOR den Inhalt der Strings nur aneinander hängt und auch die Ziffernfolgen nicht im Sinne der Arithmetik addiert - es sind ja schließlich auch keine Zahlen. Ebenfalls ist nun klar, dass man auch Strings aus bereits bestehenden Strings erstellen kann. Dies kann gerade bei immer wiederkehrenden Abläufen behilflich sein.
Ebenfalls ist es möglich, eine Ziffernfolge, die als String abgespeichert ist, in eine Zahl zu konvertieren: **variable var4 = str2num(str4)**. Dieser Befehl prüft, ob der Inhalt von String *str2* eine Zahl ist und speichert diese als *Variable* (!) mit dem Namen *var4* ab. Sollte der Inhalt keine Zahl sein wird IGOR in der History die Fehlermeldung *NaN* ausgeben, welche für „not a number“ (keine Zahl) steht.

2.3 Waves

Waves sind das zentrale Datenobjekt in Igor. Eine Wave ist ein geordneter Container für viele Zahlenwerte. Jeder Zahlenwert innerhalb einer Wave hat eine Nummer, ähnlich einem Element in einer Tabellenspalte. Die Zahlenwerte einer Wave können in einer Tabelle dargestellt werden.

Eine Wave kann auf verschieden Wegen erzeugt werden. Man sie kann aus Messdaten, die z.B. in einer Excel Tabelle stehen, in das Programm laden, welches über den Menüpunkt **Data → Load Wave → Load Excel File** zu erreichen ist.

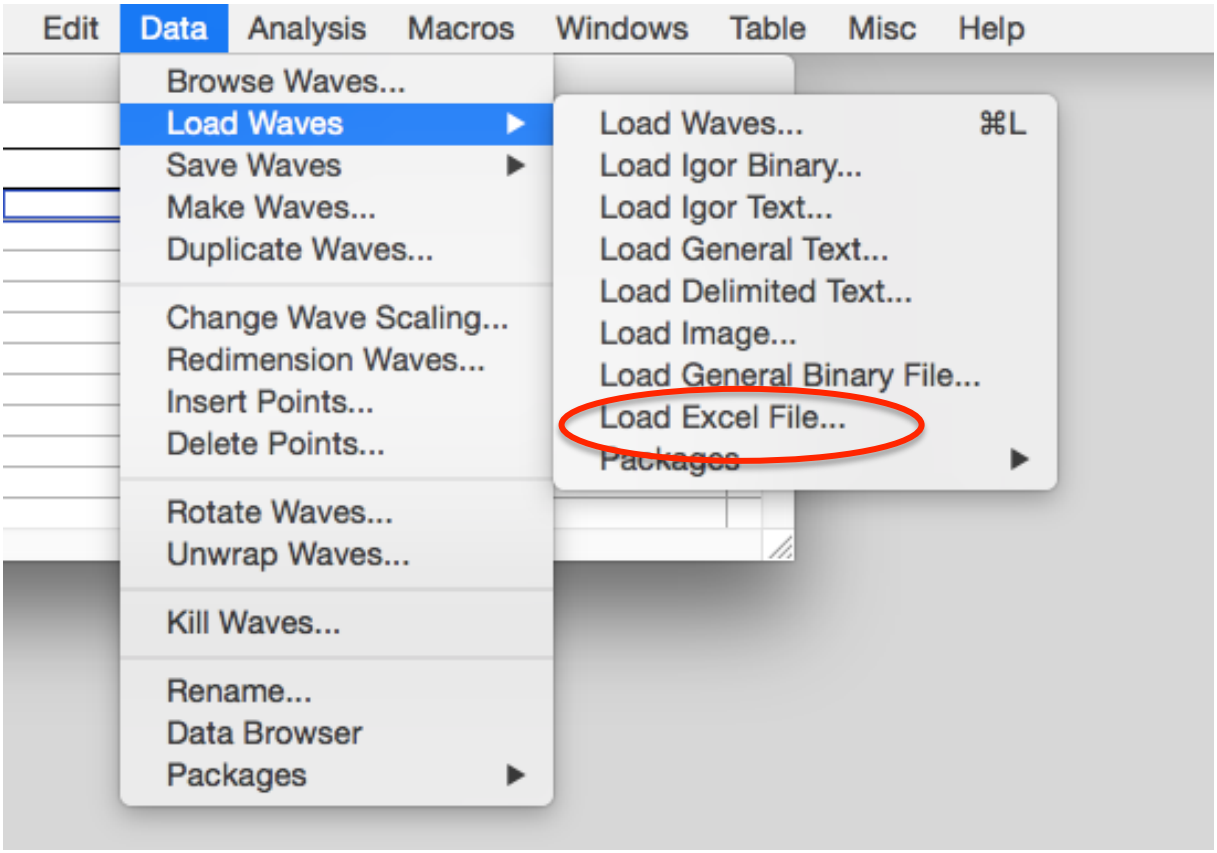

Abbildung 2: Daten aus Excel-Datei laden

Es ist auch möglich im Tabellenfenster einfach in eine neue Spalte zu klicken und dort dann Daten manuell einzutragen. Die dritte Möglichkeit, die hier aufgeführt werden soll, ist über das Kommandofenster. Mit dem Befehl **make/N=256 myWave** erstellt IGOR eine Wave mit 256 Punkten und dem Namen „myWave“, wobei die Regeln für die Namensgebung der Deklarierung mit denen für Variablen übereinstimmen.
Der Befehl

```
setscale x,0,12,myWave
```

skaliert die x-Werte der Wave myWave von 0 bis 12. Weiterhin ist es möglich, den Befehl in **Setscale /p x,0,12,myWave** zu ändern, dann ist die 12 nicht mehr das Maximum, sondern das Delta um das sich die Werte ändern. Wird **/p** zu **/I** geändert, ist 12 der Endwert. Wird keine dieser Aktionen verwendet, wird die Zahl 12 als „right value“ gesehen, dass heißt sie ist in der Skalierung der Wert den das nächste Element **nach** dem letzten Element hat.
Mit dem Befehl

```
myWave=
```

wird ein Befehl eingeleitet, welcher sich auf alle Elemente der Wave auswirkt *und die alten Werte mit neuen überschreibt*. So wird zum Beispiel mit

```
myWave= sin(x)
```

der Sinus der mit `setscale` festgelegten x-Werte in der Wave berechnet und der alte Ausgangswert mit dem Ergebnis überschrieben. Für später sei angemerkt, dass hier auch der Name einer erstellten Fitfuction eingesetzt werden kann.
Um die Wave darzustellen, nutzt man den Befehl **display myWave**. Das wichtigste Konzept ist, dass man einer Wave x-Werte zuordnen kann. IGOR hat hier 3 Befehle mit die dies auf unterschiedliche Weise tun.

Weiterhin sei angemerkt, dass man eine Wave über **killwaves myWave** gelöscht werden kann, solange sie nicht in einer Graphik oder einer Tabelle dargestellt ist. Selbiges ist ebenfalls durch einen rechtsklick auf die zu löschende Wave im Tabellenfenster möglich.